

Attention-based Extraction of Structured Information from Street View Imagery

Zbigniew Wojna* Alex Gorban† Dar-Shyang Lee† Kevin Murphy†
Qian Yu† Yeqing Li† Julian Ibarz†

* University College London

† Google Inc.

Abstract—We present a neural network model — based on CNNs, RNNs and a novel attention mechanism — which achieves 84.2% accuracy on the challenging French Street Name Signs (FSNS) dataset, significantly outperforming the previous state of the art (Smith’16), which achieved 72.46%. Furthermore, our new method is much simpler and more general than the previous approach. To demonstrate the generality of our model, we show that it also performs well on an even more challenging dataset derived from Google Street View, in which the goal is to extract business names from store fronts. Finally, we study the speed/accuracy tradeoff that results from using CNN feature extractors of different depths. Surprisingly, we find that deeper is not always better (in terms of accuracy, as well as speed). Our resulting model is simple, accurate and fast, allowing it to be used at scale on a variety of challenging real-world text extraction problems.

I. INTRODUCTION

Text recognition in an unconstrained natural environment is a challenging computer vision and machine learning problem. Traditional Optical Character Recognition (OCR) systems mainly focus on extracting text from scanned documents. Text acquired from natural scenes is more challenging due to visual artifacts, such as distortion, occlusions, directional blur, cluttered background or different viewpoints. Despite these difficulties, recent advances in deep learning have made significant progress on this problem [1]–[6].

In this paper, we concentrate not just on transcribing all the text in a given image, but instead on the harder problem of extracting a subset of useful pieces of information. The model has to focus on the important parts of the scene and to ignore visual clutter. We propose a model which leverages convolutional neural networks (CNNs), recurrent neural networks (RNNs), and a new form of spatial attention.

We benchmark our model on the French Street Name Signs dataset (FSNS) [7], derived from Google Street View. The dataset contains over 1M labeled images of visual text “in the wild”; this is significantly more than COCO Text [5], which only includes 63k labeled images. We achieve 84.2% accuracy on FSNS, significantly outperforming the previous state-of-the-art [7], which achieved 72.46%.

In addition to being more accurate, our model is simpler and makes fewer assumptions about the data, than the previous SOTA method on FSNS [7]. To demonstrate its broad applicability, we also evaluate our new model on the Street View Business Names dataset [8], showing strong results.

Finally, we study the accuracy and speed of using 3 different CNN-based feature extractors (namely inception-v2 [9], inception-v3 [10] and inception-resnet-v2 [10]) as input to our attention model. We find that inception-v3 and inception-resnet-v2 perform comparably, and both significantly outperform inception-v2. Motivated by the need for speed, we also study the effect of using “ablated” versions of these models, which use fewer layers. Interestingly, we find that for all three networks, the accuracy initially increases with depth, but then starts to decrease. This is in contrast to models trained on the ILSVRC Imagenet dataset [11], which is comparable in size to FSNS. For image classification, accuracy tends to increase with depth monotonically. We believe the difference is that image classification needs very complicated features, which are spatially invariant, whereas, for text extraction, it hurts to use to use such features.

In summary, our contributions are as follows: (1) We present a novel attention-based text reading architecture, trained in an end-to-end way, that beats the previous SOTA on the FSNS dataset by a significant margin while being considerably simpler and more general. (2) We show how our new model also gives excellent results on a newer, even more challenging, Street View dataset. (3) We study the speed/accuracy tradeoff that results from using CNNs of different depths and recommend a configuration that is accurate and efficient.

The source code and a trained model are available at github.com/tensorflow/models/tree/master/attention_ocr.

II. METHODS

In this section, we describe our model, which processes the image through a CNN, and then passes the (attentionally weighted) features into an RNN. See Figure 1 for a high level summary.

A. CNN-based feature extraction

We consider 3 kinds of CNN: inception-v2 [9], inception-v3 [10] and inception-resnet-v2 [10], which combines inception with resnets [12]. These models achieve state-of-the-art performance on the Imagenet classification challenge [11], [13].

It has been shown in [14] that features from lower layers of CNNs trained on Imagenet transfer well to other tasks. However, this still leaves open the question of which layer to use as our feature representation. We study this in Section IV-B.

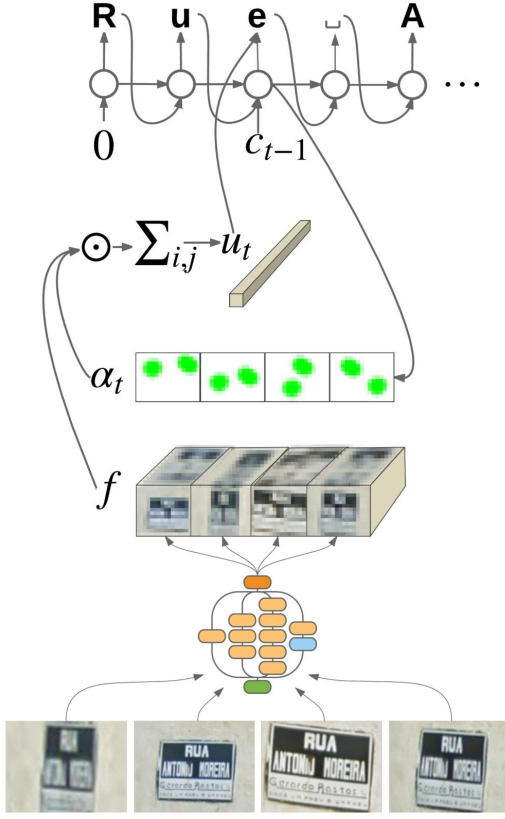


Fig. 1: Architecture of our model. We pass each of the four views through the same CNN feature extractor, and then concatenate the results into a single large feature map, shown by the cube labeled “f”. We take a spatially weighted combination to create a fixed-sized feature vector u_t , which is fed into the RNN.

(We also compared the effects of pre-training on Imagenet vs training from random initialization, and found no notable difference, so we only report results using the latter.)

We will use $f = \{f_{i,j,c}\}$ to denote the feature map derived by passing the image x through a CNN (here i, j index locations in the feature map, and c indexes channels).

B. RNN

The main challenge is how to convert the feature maps into a single text string. Following previous work, we use an RNN (specifically, an LSTM [15]) for this; this acts as a character-level language model, which takes inputs from the image, as we explain below.

Let s_t be the hidden state of the RNN at time t . The input to the RNN is determined by a spatially weighted combination of image features. This spatial attention mask is denoted by $\alpha_t = \{\alpha_{t,i,j}\}$; we explain how to compute this in Section II-C. Once we have computed the spatial mask, we compute a weighted combination of the features (the context) as follows:

$$u_{t,c} = \sum_{i,j} \alpha_{t,i,j} f_{i,j,c} \quad (1)$$

The total input to the RNN at time t is defined as

$$\hat{x}_{t,c} = W_c c_{t-1} + W_{u_1} u_{t-1} \quad (2)$$

where c_{t-1} is the one-hot encoding of the previous letter (ground truth during training, predicted during test time). We then compute the output and next state of the RNN as follows:

$$(o_t, s_t) = \text{RNNstep}(\hat{x}_{t,:}, s_{t-1}) \quad (3)$$

The final predicted distribution over letters at time t is given by

$$\hat{o}_t = \text{softmax}(W_o o_t + W_{u_2} u_t) \quad (4)$$

This combines information from the RNN, o_t , with information from the attentional feature vector, u_t . Finally, we compute the most likely letter:

$$c_t = \arg \max_c \hat{o}_t(c) \quad (5)$$

This is called greedy decoding.

C. Spatial attention

Most prior works that use spatial attention for OCR (e.g., [1], [16]–[18]) predict the mask based on the current RNN state, as follows:

$$\alpha_{t,i,j} = \text{softmax}(V_a \circ \tanh(W_s s_t + W_f f_{i,j,:})) \quad (6)$$

where \circ is elementwise multiplication, and \tanh is applied elementwise to its vector argument. This combines content from the image, via $W_f f$, with a time-varying offset, via $W_s s_t$, to determine where to look. We will use this as our baseline attention method.

The above attention mechanism is permutation invariant, meaning we could shuffle the order of the pixels and the mapping from f to α_t would remain the same (since it is applied elementwise to each location). To make the model “location aware”, we concatenate $f_{i,j,:}$ with a one-hot encoding of the spatial coordinates (i, j) , as shown in Figure 2. More precisely, we replace the argument to the \tanh function with the following:

$$W_s s_t + W_{f_1} f_{i,j,:} + W_{f_2} e_i + W_{f_3} e_j \quad (7)$$

where e_i is a one-hot encoding of coordinate i , and similarly for e_j . This is equivalent to adding a spatially varying matrix of bias terms.

Our proposal is different than the location-aware attention mechanism proposed in [19]. They suggested adding $W_{a_2} * \alpha_{t-1}$ as input to the \tanh function in Equation 6, where W_{a_2} is a convolution kernel. However, in multiline text recognition problems, we sometimes have to make a big jump to the left side of the line below (see Figure 6 for example), which cannot be captured by this approach. (We tried their approach, and it did not give good results, details omitted for brevity.)

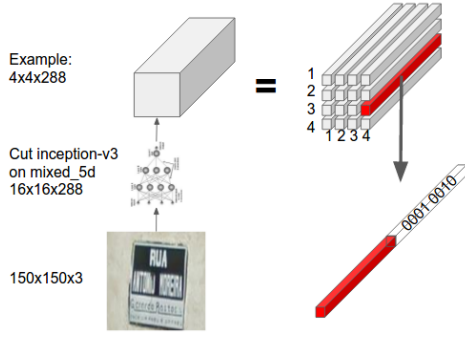


Fig. 2: Adding pixel coordinates to image features.

D. Handling multiple views

In the FSNS dataset, we have four views for each input sign, each of size 150x150. We process each of these independently, through the same CNN-based feature extractor (parameters are shared), to compute four feature maps. We then concatenate these horizontally to create a single input feature map. For example, suppose the feature map for each of the four views is $16 \times 16 \times 320$; then after concatenation, the feature map $f_{i,j,c}$ will be $64 \times 16 \times 320$. (The actual spatial resolution of the feature maps varies, as we discuss in Section IV-B.)

E. Training

We train the model using (penalized) maximum likelihood estimation, that is, we maximize $\sum_{t=1}^T \log p(y_t | y_{1:t-1}, x)$, where x is the input image, y_t is the predicted label for location t , and $T = 37$ for the FSNS dataset. (If the output string is less than 37 characters, the model is required to predict a null character.) Since the model is autoregressive, we pass in the ground truth labels as history during training, as is standard [20]. Note that we do not need ground truth bounding boxes around any of the text, which makes collecting training data much easier.

Our training method allows us to use autoregressive connections easily, which is not possible when using CTC loss [21], which was used by the previous SOTA model on FSNS [7]. We find the use of such autoregressive dependencies improves accuracy by 6%, and speeds up training 2x (details omitted for brevity).

We use stochastic gradient descent optimization with initial learning rate 0.002, decay factor 0.1 after 1,200,000 steps and momentum 0.75. We finish training after 2,000,000 steps. We use the following augmentation procedure per view: We randomly crop with the requirement to cover at least 0.8 area of the original input image and new aspect ratio to be between 0.8 and 1.2. After cropping we resize it to the initial size with one randomly chosen interpolation procedure: bilinear, bicubic, area, or nearest-neighbor. Then we apply random image distortions: change of contrast, hue, brightness, and saturation. To regularize the model, we use weight decay 0.00004, label smoothing 0.9 and LSTM values clipping to 10. LSTM unit size is 256. We use a batch size of 32 with

asynchronous training on 40 machines. For inception-resnet-v2 we have used a batch size of 12 due to GPU memory limitations. It takes less than 10 hours to train a single model with the inception-v3 network. We apply Polyak averaging [22] with decay 0.9999 to derive the weights for inference.

III. DATASETS

In this section, we describe the datasets that we use in our experiments.

A. FSNS dataset

The FSNS dataset [7] contains 965,917 training images, 38,633 validation images and 42,839 test images. Each image has up to 4 tiles, intended to be a different view of the same physical street sign from Street View imagery from France. The size of every tile is 150x150 pixels. The first view represents the ground truth physical sign (as views do not always correspond to the same sign). See Figure 4 for some examples.

All the transcriptions of the street name are up to 37 characters long. (Our model takes advantage of this fact and always runs 37 steps, with an optional out-of-alphabet padded symbol.) There are 134 possible characters to choose from at each location, but most of the street names consist only of Latin letters.

B. Street View Business Names Dataset

This is an internal dataset which contains $\sim 1M$ single view images of business storefronts extracted from Google Street View imagery. See Figure 5 for some examples. The size of every image is 352x352. All transcriptions contain up to 33 symbols, with 128 characters in the vocabulary.

This dataset is significantly more challenging than FSNS, as storefronts have a lot more variation, and richer contextual information, compared to the street name signs. Moreover, the business name can be a small fraction of the entire image.

IV. EXPERIMENTAL RESULTS

In this section, we report our experimental results on various datasets.

A. Accuracy on FSNS

We use the full sequence accuracy metric to benchmark our model. It is a challenging metric, as it requires every character to agree with the ground truth. We compare the previous state-of-the-art [7] with five different versions of our model. In particular, we use three feature extractors (inception-v2, inception-v3 and inception-resnet-v2), and combine this with two attention models (standard and our novel position-dependent attention). As we see from Table I, all our methods significantly outperform the previous state-of-the-art; inception-v3 and inception-resnet-v2 give similar performance, and both significantly beat inception-v2; finally, we see that our novel location-aware attention helps by 0.9% over standard attention.

TABLE I: Accuracy on FSNS test set.

CNN	Attention	Accuracy
Smith et al. [7]	NA	72.46%
Inception-v2	Standard	80.7%
Inception-v2	Location	81.8%
Inception-v3	Standard	83.1%
Inception-v3	Location	84.0%
Inception-resnet-v2	Standard	83.3%
Inception-resnet-v2	Location	84.2%

TABLE II: Performance of different cuts of inception-v2. Mixed layers are cuts after the inception block of concatenated convolutions.

Inception-v2 layer	Size per view	Acc	ms/Image	Depth
MaxPool_3a_3x3	19x19x192	0.539	26	3
Mixed_3b	19x19x256	0.777	31	6
Mixed_3c	19x19x320	0.803	37	9
Mixed_4a	10x10x576	0.765	34	12
Mixed_4b	10x10x576	0.789	37	15
Mixed_4c	10x10x576	0.805	38	18
Mixed_4d	10x10x576	0.804	41	21
Mixed_4e	10x10x576	0.807	44	24
Mixed_5a	5x5x1024	0.791	43	27
Mixed_5b	5x5x1024	0.760	45	30
Mixed_5c	5x5x1024	0.792	47	33

B. The effect of depth on FSNS

The main computational bottleneck in our model is the CNN-based feature extractor. To see how accuracy and speed vary as a function of the depth of the CNN, we considered “cutting” standard models at different layers, and then using these as feature extractors. We evaluate the accuracy of the resulting trained model (measured as the percentage of full sequences predicted correctly), as well as the speed (measured in milliseconds per single image inference on Tesla K40 GPU). Table II shows the results for inception-v2, Table III shows the results for inception-v3, and Table IV shows the results for inception-resnet-v2. Note that these results are using standard attention, not location-aware attention.

We see that the accuracy improves for a while, and then starts to drop as the depth increases. This trend holds for all three models. We believe the reason for this is that character recognition does not benefit from the high-level features that are needed for image classification.

When comparing the different architectures, we see that inception-resnet-v2 is the most accurate (0.833), then inception-v3 (0.831), and finally inception-v2 (0.807). We chose to use inception-v3 features from the mixed-5d layer for all the other experiments in this paper, since this is almost twice as fast as the best inception-resnet-v2 cut, and has very similar accuracy.

C. Visualization of attention on FSNS

To understand the behavior of our model, we use a visualization procedure similar to one proposed by [23]. For every predicted character k , we compute the partial derivative

TABLE III: Performance of different cuts of inception-v3.

Inception-v3 layer	Size per view	Acc	ms/Image	Depth
MaxPool_3a_3x3	35x35x64	0.157	26	3
Conv2d_3b_1x1	35x35x80	0.541	31	4
Conv2d_4a_3x3	33x33x192	0.674	37	5
MaxPool_5a_3x3	16x16x192	0.712	35	5
Mixed_5b	16x16x256	0.818	29	8
Mixed_5c	16x16x288	0.816	33	11
Mixed_5d	16x16x288	0.831	36	14
Mixed_6a	7x7x768	0.822	34	17
Mixed_6b	7x7x768	0.804	37	22
Mixed_6c	7x7x768	0.822	40	27
Mixed_6d	7x7x768	0.820	42	32
Mixed_6e	7x7x768	0.826	45	37
Mixed_7a	3x3x1280	0.674	45	41
Mixed_7b	3x3x2048	0.802	48	44
Mixed_7c	3x3x2048	0.810	51	47

TABLE IV: Performance of different cuts of inception-resnet-v2. We only report results from layer Mixed_5b and above as cuts for lower layers are identical to the inception-v3 network.

Inception-resnet-v2 layer	Size per view	Acc	ms/Image	Depth
Mixed_5b	16x16x320	0.818	43	8
Mixed_5b + 5	16x16x320	0.822	57	28
Mixed_5b + 10	16x16x320	0.824	66	48
Mixed_6a	7x7x1088	0.833	70	51
Mixed_6a + 5	7x7x1088	0.827	90	71
Mixed_6a + 10	7x7x1088	0.817	125	91
Mixed_6a + 15	7x7x1088	0.829	136	111
Mixed_6a + 20	7x7x1088	0.824	158	131
Mixed_7a	3x3x2080	0.819	307	134
Mixed_7a + 5	3x3x2080	0.823	406	154
Mixed_7a + 10	3x3x2080	0.825	512	174
Conv2d_7b_1x1	3x3x1536	0.832	546	175

of its logit a_k with respect to the input image x . We can then compute the saliency of pixel (i, j) using the formula $v_{i,j} = \left\| \frac{\partial a_k}{\partial x_{i,j}} \right\|_2$, where we compute the L2 norm of the derivative across all the color channels. To get a less noisy saliency map, we create 16 slightly perturbed versions of the image (by adding a small amount of Gaussian noise) and then average the results. Additionally, we visualize the attention map α_t by upsampling with nearest-neighbor interpolation to the size of the input.

A visualization example is shown in Figure 3. The red color represents the saliency map, and the green color represents the attention map. On the first time step, the attention map highlights the letter ‘R,’ which is the start of the name. Since ‘Rue’ is the most common beginning of a street name in this dataset, the model can predict the string ‘Rue’ without paying attention to the image for the next three iterations. This phenomenon is represented by the vertical green bars, which show that the model is just paying attention to the edge of the image, which has no informative content. We also see that the saliency map is spatially diffused in the first step since any of the letters ‘R’, ‘u’ or ‘e’ give evidence in support of

TABLE V: Breakdown of error types on FSNS.

Error type	Percent
Wrong ground truth	48
Wrong / Added / Missing accent over <i>e</i>	17
Wrong single letter inside the word	9
Wrong single letter at the beginning / end of word	8
Added / Missing hyphen (-)	7
Wrong full word	6
Read from the wrong view	3
Wrong / Added / Missing accent over different letter than <i>e</i>	2

'Rue'; for the remaining three iterations, the saliency map is zero everywhere.

On the 9th step, the model correctly emits the letter 'F', which is the start of the word 'Fonds'. Note how this word is blurred out in the first view (top left image): this kind of blurring occurs on some words in FSNS, due to mistakes in the license plate or face detection systems. Consequently, the attention and saliency maps are zero for the first view (left column) when processing 'Fonds'.

D. Error Analysis on FSNS

We analyze 100 randomly sampled wrong predictions to understand the weaknesses of our model better. 48% of the "errors" are actually due to the incorrect ground truth. Table V gives a more detailed breakdown. The most common error is due to the wrong accent over the letter *e* (it should be either *é* or *è*); interestingly, this is also the most common mistake in the ground truth transcription.

In Figure 4, we show some of the test cases where our model has a different prediction than the ground truth. (In the last example, this is due to an error in the ground truth.)

E. Results on internal datasets

We now report results on a new internal dataset. Since this dataset is not public, we just show qualitative results; the purpose is to show how the very same model can be used for extracting information from many different kinds of street signs.

Figure 5 shows an example of our system applied to an image from the Business Names dataset. Notice how the name of the business, "Lambert's Tire Service", appears in two locations: at the top, and at the bottom right. The model chooses to attend to the latter, perhaps because the font is more standard, and the writing is horizontal and not "wavy".

Figure 6 shows another example. This time, we see that the attention maps scans left to right, to read "Autopartes", but has to jump down a line and all the way to the left (like the carriage return action of a mechanical typewriter) to read the second line of text, "Lubricantes Tauro".

V. CONCLUSIONS AND FUTURE WORK

We have presented an end-to-end approach for scene text recognition which gives state-of-the-art results on the challenging FSNS dataset, and an internal dataset. Our novel attention

R
u
e

d
e
s

F
o
n
d



Fig. 3: Visualization of saliency maps (in red) and attention masks (in green) on an FSNS image. The model correctly predicts the string "Rue des Fonds Gadons"; we just show the first 12 steps.



(a) Confused by font. Pred = 'Avenue Georges Frere', GT = 'Avenue General Frere'.



(b) Read text from the wrong view. Pred='Boulevard des Talus', GT='Boulevard Charles'.



(c) Confusion due to scratched letter, which looks like 'J', but model uses its prior to produce 'O'. Pred='Impasse des Jorfèvres', GT='Impasse des Orfévres'.



(d) The model has better language prior than the human annotator. Pred='Avenue des Erables', Wrong GT='Avenue des Enadles'.

Fig. 4: Examples of FSNS signs where our prediction (Pred.) differs from ground truth (GT).



Fig. 5: Left: An example image from the internal Business Names dataset. The ground truth string is "Lambert's Tire Service", which is correctly predicted by the model. This name is written on the top of the store in a "wavy" font, but also on the right-hand side (green sign) in a more standard horizontal font. Right: Visualization of the time-averaged saliency maps (in red) and attention masks (in green). Notice how they focus on the location that contains the store name.

mechanism allows us to extract structured text information by reading only the interesting parts of the whole image.

In the future, we would like to investigate more sophisticated ways of training the RNN, such as scheduled sampling [24] or hybrid ML/RL methods [25]. We would also like to extend the system to full structured extraction of business information from storefronts.

The authors would like to thank Sergio Guadarrama, Alex Lacoste, Ray Smith, Quoc Le, Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke and Sacha Arnoud for valuable discussions and TensorFlow support.

REFERENCES

- [1] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural network for scene text detection," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2529–2541, 2016.
- [2] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1457–1464.
- [3] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016. [Online]. Available: <http://dx.doi.org/10.1007/s11263-015-0823-z>
- [4] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu *et al.*, "Icdar 2015 competition on robust reading," in *Document Analysis and*

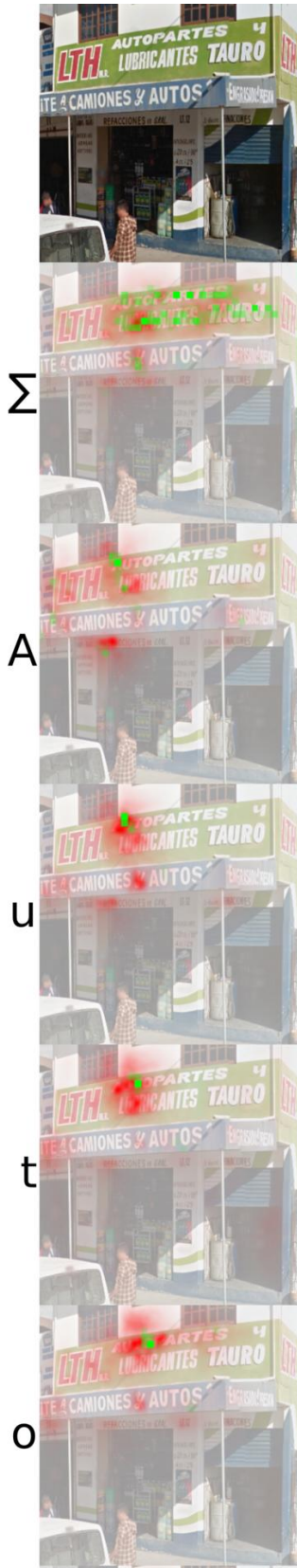


Fig. 6: Visualization of saliency maps (in red) and attention masks (in green) on a Business Names image. The model correctly predicts the string “Autopartes Lubricantes Tauro”; we just show the first few steps.

- Recognition (ICDAR), 2015 13th International Conference on. IEEE, 2015, pp. 1156–1160.
- [5] A. Veit, T. Matera, L. Neumann, J. Matas, and S. J. Belongie, “Coco-text: Dataset and benchmark for text detection and recognition in natural images,” *CoRR*, vol. abs/1601.07140, 2016. [Online]. Available: <http://arxiv.org/abs/1601.07140>
 - [6] M. Iwamura, T. Matsuda, N. Morimoto, H. Sato, Y. Ikeda, and K. Kise, *Downtown Osaka Scene Text Dataset*. Cham: Springer International Publishing, 2016, pp. 440–455. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46604-0_32
 - [7] R. Smith, C. Gu, D.-S. Lee, H. Hu, R. Unnikrishnan, J. Ibarz, S. Arnaud, and S. Lin, “End-to-end interpretation of the french street name signs dataset,” in *European Conference on Computer Vision*. Springer, 2016, pp. 411–426.
 - [8] Q. Yu, C. Szegedy, M. C. Stumpe, L. Yatziv, V. Shet, J. Ibarz, and S. Arnaud, “Large scale business discovery from street level imagery,” *arXiv preprint arXiv:1512.05430*, 2015.
 - [9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Re-thinking the inception architecture for computer vision,” *arXiv preprint arXiv:1512.00567*, 2015.
 - [10] C. Szegedy, S. Ioffe, and V. Vanhoucke, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *arXiv preprint arXiv:1602.07261*, 2016.
 - [11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “Imagenet large scale visual recognition challenge,” *Intl. J. Comp. Vision*, vol. 115, no. 3, pp. 211–252, 2015.
 - [12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
 - [13] A. Canziani, A. Paszke, and E. Culurciello, “An analysis of deep neural network models for practical applications,” *arXiv preprint arXiv:1605.07678*, 2016.
 - [14] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” in *Advances in neural information processing systems*, 2014, pp. 3320–3328.
 - [15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
 - [16] C.-Y. Lee and S. Osindero, “Recursive recurrent nets with attention modeling for ocr in the wild,” *arXiv preprint arXiv:1603.03101*, 2016.
 - [17] 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.
 - [18] B. Shi, X. Wang, P. Lv, C. Yao, and X. Bai, “Robust scene text recognition with automatic rectification,” *arXiv preprint arXiv:1603.03915*, 2016.
 - [19] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Advances in Neural Information Processing Systems*, 2015, pp. 577–585.
 - [20] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *NIPS*, 2014. [Online]. Available: <http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf>
 - [21] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.
 - [22] B. T. Polyak and A. B. Juditsky, “Acceleration of stochastic approximation by averaging,” *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, 1992.
 - [23] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” *arXiv preprint arXiv:1312.6034*, 2013.
 - [24] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” in *NIPS*, 2015. [Online]. Available: <http://arxiv.org/abs/1506.03099>
 - [25] M. Norouzi, S. Bengio, Z. Chen, N. Jaitly, M. Schuster, Y. Wu, and D. Schuurmans, “Reward augmented maximum likelihood for neural structured prediction,” in *NIPS*, 2016. [Online]. Available: <http://arxiv.org/abs/1609.00150>