What You Get Is What You See:
A Visual Markup Decompiler

Yuntian Deng
Harvard University
dengyuntian@gmail.com

Anssi Kanervisto
University of Eastern Finland
anssk@student.uef.fi

Alexander M. Rush
Harvard University
srush@seas.harvard.edu

Abstract

Building on recent advances in image caption generation and optical character recognition (OCR), we present a general-purpose, deep learning-based system to *decompile* an image into presentational markup. While this task is a well-studied problem in OCR, our method takes an inherently different, data-driven approach. Our model does not require any knowledge of the underlying markup language, and is simply trained end-to-end on real-world example data. The model employs a convolutional network for text and layout recognition in tandem with an attention-based neural machine translation system. To train and evaluate the model, we introduce a new dataset of real-world rendered mathematical expressions paired with LaTeX markup, as well as a synthetic dataset of web pages paired with HTML snippets. Experimental results show that the system is surprisingly effective at generating accurate markup for both datasets. While a standard domain-specific LaTeX OCR system achieves around 25% accuracy, our model reproduces the exact rendered image on 75% of examples.

Introduction

Optical character recognition (OCR) is most commonly used to recognize natural language from an image; however, as early as the work of (Anderson 1967), there has been research interest in converting images into structured language or *markup* that defines both the text itself and its presentational semantics. The primary focus of this work is OCR for mathematical expressions, and how to handle presentational aspects such as sub and superscript notation, special symbols, and nested fractions (Belaid and Haton 1984; Chan and Yeung 2000). The most effective systems combine specialized character segmentation with grammars of the underlying mathematical layout language (Miller and Viola 1998). A prime example of this approach is the IN-FTY system that is used to convert printed mathematical expressions to LaTeX and other markup formats (Suzuki et al. 2003).

Problems like OCR that require joint processing of image and text data have recently seen increased research interest due to the refinement of deep neural models in these two domains. For instance, advances have been made in the areas of handwriting recognition (Ciresan et al. 2010), OCR in natural scenes (Jaderberg et al. 2015; 2016; Wang et al. 2012) and image caption generation (Karpathy and Fei-Fei 2015; Vinyals et al. 2015b). At a high-level, each of these systems learns an abstract encoded representation of the input image which is then decoded to generate a textual output. In addition to performing quite well on standard tasks, these models are entirely data driven, which makes them adaptable to a wide range of datasets without requiring heavily preprocessing inputs or domain specific engineering.

The turn towards data-driven neural methods for image and text leads us to revisit the problem of generating structured markup. We consider whether a supervised model can learn to produce correct presentational markup from an image, without requiring a textual or visual grammar of the underlying markup language. While results from language modeling suggest that neural models can consistently generate syntactically correct markup (Karpathy, Johnson, and Li 2015; Vinyals et al. 2015a), it is unclear whether the full solution can be learned from markup-image pairs.

Our model, WYGIWYS [What you get is what you see], is a simple extension of the attention-based encoder-decoder model (Bahdanau, Cho, and Bengio 2014), which is now standard for machine translation. Similar to work in image captioning (Xu et al. 2015), the model incorporates a multi-layer convolutional network over the image with an attention-based recurrent neural network decoder. To adapt this model to the OCR problem and capture the document's temporal layout, we also incorporate a new source encoder layer in the form of a multi-row recurrent model applied before the application of attention. The use of attention additionally provides an alignment from the generated markup to the original source image (see Figure 1).

We also introduce two new datasets for the image-to-markup task. The preliminary experiments use a dataset of small synthetic geometric HTML examples rendered as web pages. For the main experiments, we introduce a new dataset, IM2LATEX-100K, that consists of a large collection of rendered real-world mathematical expressions collected from published articles[1]. We will publicly release this dataset as part of this work. The same model archi-

---

[1]This dataset is based on the challenge originally proposed as an OpenAI Request for Research under the title Im2Latex.
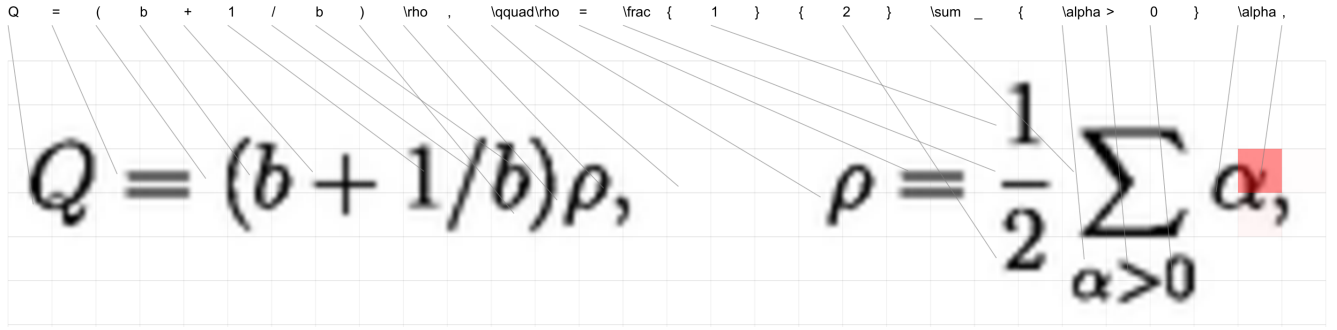
Figure 1: Example of the model generating mathematical markup. The model generates one LaTeX symbol y at a time based on the input image $\mathbf{x}$. The gray lines highlight $H' \times V'$ grid features after the CNN $\mathbf{V}$ and RNN Encoder $\tilde{\mathbf{V}}$. The dotted lines indicate the center of mass of α for each word (only non-structural words are shown). Red cells indicate the relative attention for the last token. See `http://lstm.seas.harvard.edu/latex/` for a complete interactive version of this visualization over the test set.

tecture is trained to generate HTML and LaTeX markup with the goal rendering to the exact source image. Experiments compare the output of the model with several research and commercial baselines, as well as ablations of the model. The full system for mathematical expression generation is able to match images within 15% of image edit distance, and is identical on more than 75% of real-world test examples. Additionally the use of a multi-row encoder leads to a significant increase in performance. All data, models, and evaluation scripts are publicly available at `http://lstm.seas.harvard.edu/latex/`.

Problem: Image-to-Markup Generation

We define the image-to-markup problem as converting a rendered source image to target presentational markup that fully describes both its content and layout. The source, $\mathbf{x} \in \mathcal{X}$, consists of an image with height H and width W, e.g. $\mathbb{R}^{H \times W}$ for grayscale inputs. The target, $\mathbf{y} \in \mathcal{Y}$, consists of a sequence of tokens $y_1, y_2, \cdots, y_C$ where C is the length of the output, and each y is a token in the markup language with vocabulary Σ. The rendering is defined by a possibly unknown, many-to-one, compile function, compile : $\mathcal{Y} \to \mathcal{X}$. In practice this function may be quite complicated, e.g a browser, or ill-specified, e.g. the LaTeX language.

The supervised task is to learn to approximately invert the compile function using supervised examples of its behavior. We assume that we are given instances $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), ..., (\mathbf{x}^{(J)}, \mathbf{y}^{(J)})$, with possibly differing dimensions H, W, C and that, compile$(\mathbf{y}) \approx \mathbf{x}$, for all training pairs $(\mathbf{x}, \mathbf{y})$ (assuming possible noise).

At test time, the system is given a raw input $\mathbf{x}$ rendered from ground-truth $\mathbf{y}$. It generates a hypothesis $\hat{\mathbf{y}}$ that can then be rendered by the black-box function $\hat{\mathbf{x}} = $ compile$(\hat{\mathbf{y}})$. Evaluation is done between $\hat{\mathbf{x}}$ and $\mathbf{x}$, i.e. the aim is to produce similar rendered images while $\hat{\mathbf{y}}$ may or may not be similar to the ground-truth markup $\mathbf{y}$.

Model

Our model WYGIWYS for this task combines several standard neural components from vision and natural language

processing. It first extracts image features using a convolutional neural network (CNN) and arranges the features in a grid. Each row is then *encoded* using a recurrent neural network (RNN). These encoded features are then used by an RNN decoder with a visual attention mechanism. The decoder implements a conditional language model over the vocabulary Σ, and the whole model is trained to maximize the likelihood of the observed markup. The full structure is illustrated in Figure 2. We describe the model in more detail.

Convolutional Network   The visual features of an image are extracted with a multi-layer convolutional neural network interleaved with max-pooling layers. This network architecture is now standard; we model it specifically after the network used by Shi et al. (2015) for OCR from images (specification is given in Table 2.) Unlike some recent OCR work (Jaderberg et al. 2015; Lee and Osindero 2016), we do not use final fully-connected layers (Ioffe and Szegedy 2015), since we want to preserve the locality of CNN features in order to use visual attention. The CNN takes the raw input $\mathbb{R}^{H \times W}$ and produces a feature grid $\mathbf{V}$ of size $D \times H' \times W'$, where c denotes the number of channels and H' and W' are the reduced sizes from pooling.

Row Encoder   In attention-based image captioning (Xu et al. 2015), the image feature grid can be directly fed into the decoder. For OCR, the visual features fed in to the decoder contain significant relative sequential order information. Therefore we experiment with an additional RNN encoder module that re-encodes each row of the grid. Intuitively, we expect this to help in two ways: (1) many markup languages default to a left-to-right order, which can be easily learned by the encoder, (2) the RNN can utilize the surrounding horizontal context to refine the hidden representation.

Formally, a recurrent neural network (RNN) is a parameterized function $\mathbf{RNN}$ that recursively maps an input vector and a hidden state to a new hidden state. At time t, the hidden state is updated with an input $\mathbf{v}_t$ in the following manner: $\mathbf{h}_t = \mathbf{RNN}(\mathbf{h}_{t-1}, \mathbf{v}_t; \theta)$ and $\mathbf{h}_0$ is an initial state. In practice there are many different variants of $\mathbf{RNN}$; however, long short-term memory networks (LSTMs) (Hochreiter and

Schmidhuber 1997) have been shown to be very effective for most NLP tasks. For simplicity we will describe the model as an RNN, but all experiments use LSTM networks.

In this model, the new feature grid $\tilde{\mathbf{V}}$ is created from $\mathbf{V}$ by running an RNN across each row of that input. Recursively for all rows $h \in \{1, \ldots, H'\}$ and columns $w \in \{1, \ldots, W'\}$, the new features are defined as $\tilde{\mathbf{V}}_{h,w} = \mathbf{RNN}(\tilde{\mathbf{V}}_{h,w-1}, \mathbf{V}_{h,w})$. In order to capture the sequential order information in vertical direction, we use a trainable initial hidden state $\tilde{\mathbf{V}}_{h,0}$ for each row, which we name positional embeddings.

Decoder The target markup tokens $\{y_t\}$ are then generated by a decoder based on a sequence of annotation grid $\tilde{\mathbf{V}}$. The decoder is trained as a conditional language model to give the probability of the next token given the history and the annotations.

This language model is defined on top of a decoder RNN,

$$p(y_{t+1}|y_1, \ldots, y_t, \tilde{\mathbf{V}}) = \text{softmax}(\mathbf{W}^{out}\mathbf{o}_t)$$

where $\mathbf{W}^{out}$ is a learned linear transformation and $\mathbf{o}_t = \tanh(\mathbf{W}^c[\mathbf{h}_t; \mathbf{c}_t])$. The vector $\mathbf{h}_t$ is used to summarize the decoding history: $\mathbf{h}_t = \mathbf{RNN}(\mathbf{h}_{t-1}, [y_{t-1}; \mathbf{o}_{t-1}])$. The context vector $\mathbf{c}_t$, defined below, is used to capture the context information from the annotation grid.

At each time step t, the context vector $\mathbf{c}_t$ takes into account the annotation grid. However, since most annotation cells are likely irrelevant, the model should know which cells to attend to t. We use an *attention model* $a(\cdot)$ to model this alignment (Bahdanau, Cho, and Bengio 2014):

$$\mathbf{e}_t = a(\mathbf{h}_t, \{\tilde{\mathbf{V}}_{h,w}\})$$
$$\alpha_t = \text{softmax}(\mathbf{e}_t)$$
$$\mathbf{c}_t = \phi(\{\tilde{\mathbf{V}}_{h,w}\}, \alpha_t)$$

where α_t are the weights calculated based on $\mathbf{e}_t$, and the weight vector α_t and all annotation vectors $\{\tilde{\mathbf{V}}_t\}$ are combined to form a *context vector* $\mathbf{c}_t$. Note there are different choices for a and ϕ, we follow past empirical work and use $e_{it} = \beta^T\tanh(W_h\mathbf{h}_{i-1} + W_v\tilde{\mathbf{v}}_t)$ and $\mathbf{c}_i = \sum_t \alpha_{it}\mathbf{v}_t$ (Luong, Pham, and Manning 2015). The vectors $\mathbf{c}_t$ and $\mathbf{h}_t$ are concatenated together to predict the probability of the token y_t. Figure 1 shows a real example of the attention distribution α_t at each step of the model.

Training and Generation The complete model is trained end-to-end to maximize the likelihood of the observed training data. Beyond the training data, the model is given no other information about the markup language or the generating process. To generate markup from unseen images, we simply use beam search at test time with no further hard constraints.

Data

While there are some datasets available for the image-to-markup generation problem (Mouchere et al. 2012; 2013;
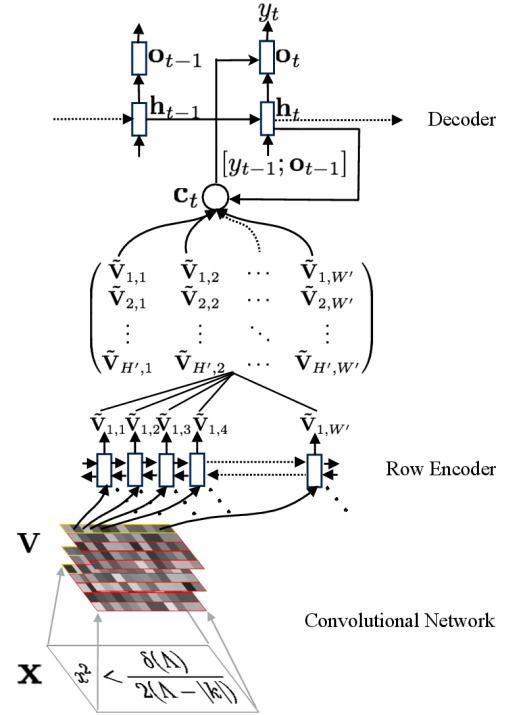

Figure 2: Network structure of WYGIWYS. Given an input image, a CNN is applied to extract visual features, then for each row in the final feature map we employ an RNN encoder. The encoded features are then used by an RNN decoder with a visual attention mechanism to produce final outputs. For clarity we only show the RNN encoding at the first row and the decoding at time step t.

Lin et al. 2012), they are all too small for training a data-driven system. We therefore introduce two new datasets for this task: a preliminary web page dataset using snippets of HTML, and a large real-world mathematical expression dataset in LaTeX.

Web Page-to-HTML

Our preliminary dataset is a synthetically generated collection of "web pages", used as a test for whether the model can learn relative spatial positioning. We generate a dataset consisting of 100k unique HTML snippets and the corresponding rendered images. The images are compiled using Webkit yielding images of size 100×100. The task is to infer the HTML markup based on the rendered image.

The HTML markup is generated using a simple context-free grammar. The grammar recursively generates `div`'s each with a solid border, a random width, and a random float (left or right). Each `div` can optionally be recursively split vertically or horizontally. [2] The maximum depth of the recursion is limited to 2 nested layers of `div`'s. Finally we impose several constraints to the widths of the elements to

---

[2] Additionally each `div` contains a randomly generated span of characters of a random font size.
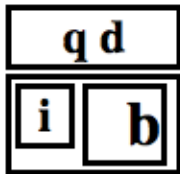
```
<div style=' margin:2px; border:
solid; text-align:center; ' > <
span style=' font-weight:bold;
text-align:left; font-size:150%; '
 > q d</span> </div> <div style='
margin:2px; border:solid; ' > ...
```

Figure 3: An example web page image and its associated HTML snippet.

avoid overlaps. Figure 3 shows an example snippet sampled from the dataset.

Math-to-LaTeX

Our primary dataset, IM2LATEX-100K, collects a large-corpus of real-world mathematical expressions written in LaTeX. This dataset provides a much more difficult test-bed for learning how to reproduce naturally occurring markup.

Corpus The IM2LATEX-100K dataset provides 103,556 different LaTeX math equations along with rendered pictures. We extract formulas by parsing LaTeX sources of papers from the arXiv. LaTeX sources are obtained from tasks I and II of the 2003 KDD cup (Gehrke, Ginsparg, and Kleinberg 2003) containing over 60,000 papers.

To extract formulas from the LaTex sources, we match the Latex sources with simple regular expressions, namely \begin{equation} (.*?) \end{equation}, $$ (.*?) $$, \[(.*?) \] and \((.*?) \). We only keep matches whose total number of characters fall in the range from 40 to 1024 to avoid single symbols, large matrices or text sentences. With these settings we extract over 800,000 different LaTeX formulas. Out of the remaining formulas, around 100 thousand are rendered in a vanilla LaTex environment. Rendering is fulfilled with *pdflatex*[3] and formulas that fail to compile are excluded. Then the rendered PDF files are converted to PNG format[4]. The final dataset we provide contains 103,556 images of resolution 1654×2339 with black equations against transparent background, and the corresponding LaTeX formulas.

The dataset is separated into training set (83,883 equations), validation set (9,319 equations) and test set (10,354 equations) for standardized experiment setup. The LaTeX formulas range from 38 to 997 characters with mean 118 and median 98.

Tokenization Training the model requires settling on a token set Σ. One option is to use a purely character-based model. While this method requires fewer assumptions, character-based NMT models have been less effective, slower, and significantly more memory intensive than word-based models. Therefore original markup is simply split into

---

[3]LaTeX (version 3.1415926-2.5-1.40.14)

[4]We use the *ImageMagick* convert tool with parameters -density 200 -quality 100

Transform	Original	Normalized
SubSup	H^I_1	H_I^1
ReqBrack	H^I	H^{I}
Desugar	H'	H^{\prime}
ExpOperators	\sin	\operatorname{sin}
InfixPrefix	\over	\frac{}{}
MatrixEnv	\matrix	\begin{array}...
Drop	\label{}	–

Table 1: Preprocessing transformations applied to LaTeX abstract syntax tree in normalizing mode. These transformations are mostly safe, although there are some corner cases where they lead to small differences in output.

Conv	Pool
c:512, k:(3,3), s:(1,1), p:(0,0), bn	-
c:512, k:(3,3), s:(1,1), p:(1,1), bn	po:(1,2), s:(1,2), p:(0,0)
c:256, k:(3,3), s:(1,1), p:(1,1)	po:(2,1), s:(2,1), p(0,0)
c:256, k:(3,3), s:(1,1), p:(1,1), bn	-
c:128, k:(3,3), s:(1,1), p:(1,1)	po:(2,2), s:(2,2), p:(0,0)
c:64, k:(3,3), s:(1,1), p:(1,1)	po:(2,2), s:(2,2), p(2,2)

Table 2: CNN specification. 'Conv': convolution layer, 'Pool: max-pooling layer. 'c': number of filters, 'k': kernel size, 's': stride size, 'p': padding size, 'po': , 'bn': with batch normalization. The sizes are in order (height, width).

minimal meaningful LaTeX tokens, e.g. for observed characters, symbols such as \sigma, modifier characters such as ^, functions, accents, environments, brackets and other miscellaneous commands.

Optional: Normalization Finally we note that naturally occurring LaTeX contains many different expressions that produce identical output. We therefore experiment with an optional normalization preprocessing step to eliminate spurious ambiguity (prior to training). For normalization, we wrote a LaTeX parser[5] to convert the markup to an abstract syntax tree. We then apply a set of safe normalizing tree transformation to eliminate common spurious ambiguity, shown in Table 1. Note this only changes the training data, and does not change the model itself.

Experimental Setup

To test this approach we compare the proposed model to several other classical OCR baselines, neural models, and ablations, on the HTML and LaTeX decompiling tasks.

Baselines

The current best-available OCR-based mathematical expression recognition system is the InftyReader system, a proprietary commercial implementation of the INFTY system of (Suzuki et al. 2003). This system combines symbol recognition and structural analysis phases. Additionally we experimented with the open-source AbiWord OCR system, which

---

[5]Based on KaTeX parser https://khan.github.io/KaTeX/

Model	Preprocessing	BLEU (tok)	BLEU (norm)	Edit Distance	Exact Match	Exact Match (-ws)
INFTY	-	51.20	66.65	53.82	15.60	26.66
CNNENC	norm	52.53	75.01	61.17	53.53	55.72
WYGIWYS	tok	73.71	73.97	84.26	74.46	77.04
WYGIWYS	norm	58.41	87.73	87.60	77.46	79.88

Table 3: Main experimental results on the IM2LATEX-100K dataset. Reports the BLEU score compared to the tokenized formulas (BLEU (tok)), and the BLEU score compared to the normalized formulas (BLEU (norm)), column-wise image edit distance, exact match, and exact match without whitespace columns.

contains a Tex generation mode (Wen 2002). However we found that this system performed too poorly on this task to compare.

For a neural model, a natural comparison is to standard image captioning approaches (Xu et al. 2015). As our model is based on this approach, we compare by removing the encoder RNN, i.e. replacing $\tilde{V}$ with V, and increasing the number of CNN such that the number of parameters is approximately the same. We name this mode textscCNNEnc.

We also run experiments comparing the model to traditional LM approaches, including a standard NGRAM model (5-gram trained with Kneser-Ney smoothing) and an LSTM-LM. This indicates how much of the improvement comes from improving language modeling of the underlying markup. Finally for LaTeX we also evaluate with full normalization, norm, versus simple tokenization, tok.

Evaluation

Our core evaluation method is to check the accuracy of the rendered markup output image $\hat{x}$ compared to the true image x. The main evaluation reports the column-wise edit distance between the gold and predicted images. Explicitly we discretize generated columns, and compare the edit distance sequences. The final score is the total number of edit distance ops used divided by the maximum number in the dataset. Additionally we check the exact match accuracy with the original image as well as the value after eliminating whitespace columns.[6] We also include standard intrinsic text generation metrics, conditional language model perplexity and BLEU score (Papineni et al. 2002). Note that both of these metrics are sensitive to the fact that the markup languages have spurious ambiguity, so a deterministic perplexity of 1 would be impossible.

Implementation Details

The same model and hyperparameters are used for both the image-to-markup tasks. The CNN specifications are summarized in Table 2. The model uses single-layers LSTMs for all RNNs. We use a bi-directional RNN for the encoder. The hidden state of the encoder RNN is of size 256, decoder RNN of 512, and token embeddings of size 80. The model has 9.48 million parameters in total. We use mini-batch stochastic gradient descent to learn the parameters.

Model	Train Perp	Test Perp	Exact Match
NGRAM	1.60	1.60	-
WYGIWYS	1.06	1.06	97.61

Table 4: Web Page-to-HTML results. Reports the training perplexity, test perplexity, as well as exact match score on test set.

Model	Ablation	Train Perp	Test Perp
NGRAM		5.50	8.95
LSTM-LM	-RNNEnc-CNN	4.13	5.22
CNNENC	-RNNEnc	1.08	1.18
WYGIWYS	-PosEmbed	1.03	1.12
WYGIWYS		1.05	1.11

Table 5: Image-to-LaTeX ablation experiments. Compares simple LM approaches and versions of the full model with different encoders.

The initial learning rate is set to 0.1, and we halve it once the validation perplexity does not decrease. We train the model for 12 epochs and use the validation perplexity to choose the best model. During test phase, we use beam search with beam size 5.

The system is built using Torch (Collobert, Kavukcuoglu, and Farabet 2011) based on the Seq2seq-attn NMT system [7]. Experiments are run on a 12GB NVidia Titan X GPU.

HTML All images start as 100×100 color input, which are then down-sampled to be grayscale images of size 64×64. We then normalize pixels to the range $[-1, 1]$. During training, we only use training instances with less than 100 output tokens to accelerate the training process. The batch size is set to 100. The training process takes 4 hours.

LaTeX Original images are cropped to only the formula area, and padded with 8 pixels to the top, left, right and bottom. For efficiency we downsample all images to half of their original sizes. To facilitate batching, we group images into similar sizes and pad with whitespace.[8] All images of

---

[6]In practice we found that the LaTeX renderer often misaligns identical expressions by several pixels. To correct for this, only misalignments of ≥ 5 pixels wide are "exact" match errors.

[7]https://github.com/harvardnlp/seq2seq-attn

[8]Width-Height groups used are (120,50), (160,40), (200,40), (200,50), (240,40), (240,50), (280,40), (280,50), (320,40),
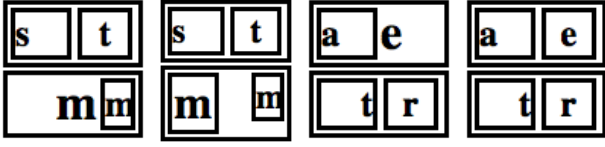
Figure 4: Typical errors in the HTML dataset. Left to right: 1st ground truth, 1st predicted, 2nd ground truth, 2nd predicted.

larger sizes, LaTeX formulas with more than 150 tokens, or cannot be parsed by our parser are ignored during training and validation (but included at test). Training batch size is set to 20 due to the size limit of GPU memory. Training process takes about 20 hours.

Results

Initial experiments on the HTML dataset are shown in Table 4. The model is able to reach a perplexity of 1.06 and an exact match accuracy of over 97.5%. These results indicate that the model is able to learn to identify and generate the correct output based on spatial cues. Much of the remaining perplexity is due to ambiguities in the underlying markup language. Typical mistakes are shown in Table 4. The few issues occur in font-size and matching relative sizes of divs.

The main experimental results on mathematical expressions are given in Table 3. These results compare several different systems on the task of decompiling rendered La-TeX. The classical INFTY system is able to do quite well in terms of text accuracy, but performs poorly on the more rigorous image metrics. Our reimplementation of the Image-to-Caption work CNNENC does much better, pushing that number to above 50%. Our full system with RNN encoder increases this value above 75%, achieving very high accuracy on this task. We expected the LaTeX normalizer to greatly increase performance, but only give a few points of accuracy gain, despite achieving high normalized BLEU. This indicates that the decoder LM is able to learn well despite the ambiguities in real-world LaTeX.

To better understand the contribution of each part of the model, we run ablation experiments removing different aspects, which are shown in Table 5. The simplest model is a standard NGRAM LM on LaTeX with a perplexity of around 8. Simply switching to an LSTM-LM reduces the value to 5, likely due to its ability to count parentheses and nesting-levels. Adding the image data with a CNN further reduces the perplexity down to 1.18. Adding the encoder LSTM adds a small gain to 1.12, but actually makes a large difference in final accuracy. Adding the positional embeddings (trainable initial states for each row) adds a tiny gain.

The main non-spacing errors of the model are shown in Table 6. These results show the most common presentation-affecting replacements of the norm model. Most of the errors again come from font issues, such as using small parentheses instead of large ones or using standard math font instead of escaping or using math cal. Figure 5 shows several ex-

---

(320,50), (360,40), (360,50), (360,60), (360, 100), (400,50), (400,160), (500,100).

\mathcal	⇒	-	\, (	⇒	\left(
\right)	⇒	)	\left[	⇒	[
\left(	⇒	(	\left(	⇒	\, (
)	⇒	\right)	\big	⇒	-
\mathbf	⇒	-	\right)	⇒	) \,
(	⇒	\left(	\prime	⇒	-
\mathrm	⇒	-	_ {	⇒	-
\right]	⇒	]	\widetilde	⇒	-
\mathrm	⇒		\bot	⇒	\perp
		\operatorname			

Table 6: Most common presentation-affecting errors in the LaTeX norm validation dataset.

Figure 5: Typical errors in the LaTeX dataset. We show the operations needed to get ground truth from the rendered predictions. Red denotes add operations and blue denotes delete operations.

amples of common errors. Typically most of the structure of the expression is preserved, but with one or two symbol recognition errors.

Conclusion and Future Work

We have presented a visual attention-based model, WYGI-WYS, for OCR of presentational markup. The model acts as a "visual decompiler" for markup such as HTML and La-TeX. We also introduce a new dataset IM2LATEX-100K that provides a test bed for this task of Image-to-Markup generation. These contributions provide a new view on the task of structured text OCR, and show data-driven models can be surprisingly effective, even without any knowledge of the underlying language.

Possible future directions for this work include: scaling the system to run on full websites or for document decompilation, using similar approach for handwritten mathematical expressions or HTML from informal sketches, or combining these methods with neural inference machines such as MemNNs (Weston, Chopra, and Bordes 2014) for more complicated markup or reference variables.

References

Anderson, R. H. 1967. Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, 436–459. ACM.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Belaid, A., and Haton, J.-P. 1984. A syntactic approach for handwritten mathematical formula recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1):105–111.

Chan, K., and Yeung, D. 2000. Mathematical expression recognition: a survey. *IJDAR* 3(1):3–15.

Ciresan, D. C.; Meier, U.; Gambardella, L. M.; and Schmidhuber, J. 2010. Deep, big, simple neural nets for handwritten digit recognition. *Neural computation* 22(12):3207–3220.

Collobert, R.; Kavukcuoglu, K.; and Farabet, C. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

Gehrke, J.; Ginsparg, P.; and Kleinberg, J. 2003. Overview of the 2003 kdd cup. *ACM SIGKDD Explorations Newsletter* 5(2):149–151.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of The 32nd International Conference on Machine Learning*, 448–456.

Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2015. Deep structured output learning for unconstrained text recognition. *ICLR*.

Jaderberg, M.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2016. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision* 116(1):1–20.

Karpathy, A., and Fei-Fei, L. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3128–3137.

Karpathy, A.; Johnson, J.; and Li, F.-F. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Lee, C.-Y., and Osindero, S. 2016. Recursive recurrent nets with attention modeling for ocr in the wild. *arXiv preprint arXiv:1603.03101*.

Lin, X.; Gao, L.; Tang, Z.; Lin, X.; and Hu, X. 2012. Performance evaluation of mathematical formula identification. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, 287–291. IEEE.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *EMNLP*.

Miller, E. G., and Viola, P. A. 1998. Ambiguity and constraint in mathematical expression recognition. In *AAAI/IAAI*, 784–791.

Mouchere, H.; Viard-Gaudin, C.; Kim, D. H.; Kim, J. H.; and Garain, U. 2012. Icfhr 2012 competition on recognition of on-line mathematical expressions (crohme 2012). In *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, 811–816. IEEE.

Mouchere, H.; Viard-Gaudin, C.; Zanibbi, R.; Garain, U.; Kim, D. H.; and Kim, J. H. 2013. Icdar 2013 crohme: Third international competition on recognition of online handwritten mathematical expressions. In *2013 12th International Conference on Document Analysis and Recognition*, 1428–1432. IEEE.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Shi, B.; Bai, X.; and Yao, C. 2015. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *arXiv preprint arXiv:1507.05717*.

Suzuki, M.; Tamari, F.; Fukuda, R.; Uchida, S.; and Kanahori, T. 2003. Infty: an integrated ocr system for mathematical documents. In *Proceedings of the 2003 ACM symposium on Document engineering*, 95–104. ACM.

Vinyals, O.; Kaiser, Ł.; Koo, T.; Petrov, S.; Sutskever, I.; and Hinton, G. 2015a. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2755–2763.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015b. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164.

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

Wen, H. 2002. Abiword: Open source's answer to microsoft word. *Linux Dev Center, downloaded from http://www. linuxdevcenter. com/lpt/a/1636* 1–3.

Weston, J.; Chopra, S.; and Bordes, A. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of The 32nd International Conference on Machine Learning*, 2048–2057.