Teaching Machines to Code:
Neural Markup Generation with Interpretable Attention

Sumeet S. Singh
Independent Researcher
Saratoga, CA 95070
sumeet@singhonline.info

Abstract

We present a neural transducer model with visual attention that learns to generate LaTeX markup of a real-world math formula given its image. Applying sequence modeling and transduction techniques that have been very successful across modalities such as natural language, image, handwriting, speech and audio; we construct an image-to-markup model that learns to produce syntactically and semantically correct LaTeX markup code over 150 words long and achieves a BLEU score of 89%; improving upon the previous state-of-art for the Im2Latex problem. We also demonstrate with heat-map visualization how attention helps in interpreting the model and can pinpoint (localize) symbols on the image accurately despite having been trained without any bounding box data.

1 Introduction

In the past decade, deep neural network models based on RNNs[1], CNNs[2] and 'attention' [?] have been shown to be very powerful sequence modelers and transducers. Their ability to model joint distributions of real-world data has been demonstrated through remarkable achievements in a broad spectrum of generative tasks such as; image synthesis [? ? ? ?], image description [? ? ? ? ?], video description [?], speech and audio synthesis [?], handwriting recognition [? ?], handwriting synthesis [?], machine translation [? ? ? ?], speech recognition [? ? ?], etc. [? ?]

One class of sequence models employ the so-called ***encoder-decoder*** [?] or ***sequence-to-sequence*** [?] architecture, wherein an *encoder* encodes a source sequence into feature vectors, which a *decoder* employs to produce the target sequence. The source and target sequences may either belong to the same modality (e.g. in machine translation use-cases) or different modalities (e.g. in image-to-text, text-to-image, speech-to-text); the encoder / decoder sub-models being constructed accordingly. The entire model is trained end-to-end using supervised-learning techniques. In recent years, this architecture has been augmented with an ***attention and alignment*** model which selects a subset of the feature vectors for decoding. It has been shown to help with longer sequences [? ?]. Among other things, this architecture has been used for image-captioning [?]. In our work we employ a encoder-decoder architecture with attention, to map images of math formulas into corresponding LaTeX markup code. The contributions of this paper are: 1) Solves the Im2Latex problem[100] and improves over the previous best reported BLEU score by 1.27% BLEU, 2) Pushes the boundaries of the neural encoder-decoder architecture with visual attention, 3) Analyses variations of the model and cost function. Specifically we note the changes to the base model [?] and what impact those had on performance, 4) Demonstrates the use of attention visualization for model interpretation and

---

[1]Recurrent Neural Network.

[2]Convolutional Neural Networks and variants such as dilated CNNs [?].

5) Demonstrates how attention can be used to localize objects (symbols) in an image despite having been trained without bounding box data.

1.1 The IM2LATEX problem

The IM2LATEX Problem is a request for research proposed by OpenAI. The challenge is to build a Neural Markup Generation model that can be trained end-to-end to generate the LaTeX markup of a math formula given its image. Data for this problem was produced by rendering single-line real-world LaTeX formulas obtained from the KDD Cup 2003 dataset. The resulting grayscale images were used as the input samples while the original markup was used as the label/target sequence. Each training/test sample (Figure 1) is comprised of an input image x and a cor-

$$S_0 = \sum_l \frac{1}{2\Delta_l^2} \operatorname{Tr} \phi_l^a \phi_{-l}^a + \sum_l \frac{1}{2\epsilon_l^2} \operatorname{Tr} f_l^a f_{-l}^a + \sum_r \frac{1}{g_r} \operatorname{Tr} \bar{\psi}_r^a \psi_r^a.$$

S _ { 0 } = \sum _ { l } \frac { 1 } { 2 \Delta _ { l } ^ { 2 } } \mathrm { T r } \, \phi _ { l } ^ { a } \phi _ { −1 } ^ { a } + \sum _ { l } \frac { 1 } { 2 \epsilon _ { l } ^ { 2 } } \mathrm { T r } \, f _ { l } ^ { a } f _ { −1 } ^ { a } + \sum _ { r } \frac { 1 } { g _ { r } } \mathrm { T r } \, \bar { \psi } _ { r } ^ { a } \psi _ { r } ^ { a } \, .

S _ { 0 } = \sum _ { l } { \frac { 1 } { 2 \Delta _ { l } ^ { 2 } } } \mathrm { T r } \, \phi _ { l } ^ { a } \phi _ { −1 } ^ { a } + \sum _ { l } { \frac { 1 } { 2 \epsilon _ { l } ^ { 2 } } } \mathrm { T r } \, f _ { l } ^ { a } f _ { −i } ^ { a } + \sum _ { r } { \frac { 1 } { g _ { r } } } \mathrm { T r } \, \psi _ { r } ^ { a } \psi _ { r } ^ { a } \, .

Figure 1: A training sample: At the top is the input image x, middle the target sequence y ($\tau = 145$) and bottom the predicted sequence $\hat{y}$ ($\tau = 148$). Each space-separated word in y and $\hat{y} \in V$

responding target LaTeX-sequence y of length τ. Each word y of the target sequence, belongs to the vocabulary of the dataset plus two special tokens: beginning-of-sequence <bos> and end-of-sequence <eos>. Denoting image dimensions as H_I, W_I and C_I and the vocabulary as a set V of K words, we represent $x \in \mathbb{R}^{H_I \times W_I \times C_I}$, $V := \{\text{LaTeX tokens}, \text{<eos>}, \text{<bos>}\}$; $|V| = K$ and $y := (y_1, \ldots, y_\tau)$; $y_t \in \{1, \ldots, K\}$. The task is to generate markup that a LaTeX compiler will render back to the original image. Therefore, our model needs to generate syntactically and semantically correct markup, by simply 'looking' at the image: i.e. it should jointly model vision and language.

2 Image to markup model

Our model (Figure 2a) has the same basic architecture as [**?**] (which we call our baseline model) in the way the encoder, decoder and a visual attention interact. However there are significant differences in the sub-models which we notate in the remainder of this paper and in the appendix.

2.1 Encoder

All images are standardized to a fixed size by centering and padding with white pixels. Then they are linearly transformed (whitened) to lie in the range [-0.5,0.5]. A deep CNN then encodes the whitened image into a *visual feature grid* $\acute{A}$, having $\acute{H} \times \acute{W}$ (i.e. height × width) *visual feature vectors* $a_{(\acute{h},\acute{w})} \in \mathbb{R}^{\acute{D}}$. The visual feature vectors are then concatenated (pooled) together in strides of shape $[S_H, S_W]$; begetting *pooled feature vectors* $a_{(h,w)} \in \mathbb{R}^D$, where $D = \acute{D} \cdot S_H \cdot S_W$. The resulting feature map A, has a correspondingly shrunken shape $[H, W]$; where $H = \acute{H}/S_H$ and $W = S_W/\acute{W}$.

$$A := \begin{bmatrix} a_{(1,1)} & \cdots & a_{(1,W)} \\ \vdots & \vdots & \vdots \\ a_{(H,1)} & \cdots & a_{(H,W)} \end{bmatrix} \quad (1)$$

Each pooled feature vector can be viewed as a rectangular window into the image, bounded by its receptive field.[3] The idea behind this is to partition the image into spatially localized regional encodings and setup a decoder architecture (Section 2.2) that selects/emphasizes only the relevant

---

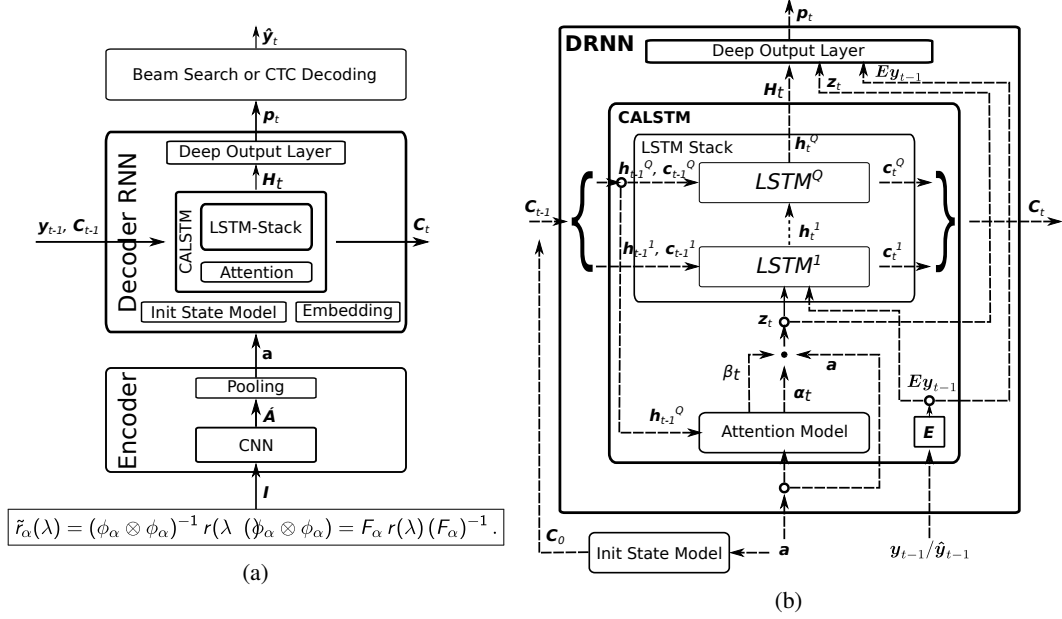[3]Neighboring regions overlap but each region is distinct overall.

Figure 2: (a) Model outline showing major parts of the model. Beam search decoder is only used during inferencing, not training. LSTM-Stack and Attention model jointly form a Conditioned Attentive LSTM stack (CALSTM) which can itself be stacked. (b) Expanded view of Decoder RNN showing its sub-models. There are three nested RNN cells in all: The decoder RNN (DRNN) at the top level, nesting the CALSTM which nests the LSTM-Stack. The Init Model does not participate in recurrence, therefore its is shown outside the box.

regions at each time-step t, while filtering-out/de-emphasizing the rest. **?** showed that such piecewise encoding enables modeling longer sequences as opposed to models that encode the entire input into a single feature vector [**?** **?**]. [4] Pooling allows us to construct encoders with different receptive field sizes. We share results of two such models: I2L-NOPOOL with no feature pooling and pooled feature grid shape [4,34] and I2L-STRIPS having stride [4,1] and pooled feature grid shape [1,34]. Finally, for convenience we represent A as a flattened sequence a (Equation 2). See the appendix for more details.

$$a := (a_1, \ldots, a_L); \; a_l \in \mathbb{R}^D; \; l = H(h-1) + w; \; L = HW \qquad (2)$$

2.2 Decoder

The decoder is a language modeler and generator. It is a Recurrent Neural Network (DRNN in Figure 2b) that models the discrete probably distribution p_t, of the output word y_t, conditioned on the sequence of previous words $y_{<t}$ and relevant regions of the encoded image a[5] (Equations 3). Probability of the entire output sequence y given image a is therefore given by Equation 4.

$$
\begin{aligned}
p_t &: \; \{1, \ldots, K\} \to [0,1] \\
y_t &\sim \; p_t \\
p_t(y_t) &:= \; Pr(y_t | y_{<t}, a) \qquad (3)
\end{aligned}
$$

$$Pr(y|a) = \prod_{t=1}^{\tau} p_t(y_t) \qquad (4)$$

The DRNN receives the previous word y_{t-1} and encoded image a as inputs. In addition, it maintains an internal state C_t that propagates information (features) extracted from an initial state, the output

---

[4]That said, **?** employ a bidirectional-LSTM [**?**] encoder whose receptive field does encompass the entire input anyway! (Although that does not necessarily mean that the bi-LSTM will encode the entire image). Likewise **?** who also solve the IM2LATEX problem also employ a bi-directional LSTM stacked on top of a CNN-encoder in order to get full view of the image. In contrast, our visual feature vectors hold only spatially local information which we found are sufficient to achieve good accuracy. This is probably owing to the nature of the problem; i.e. transcribing a one-line math formula into LATEXsequence requires only local information at each step.

[5]This is now a very standard way to model sequence (sentence) probabilities in neural sequence-generators. See [**?**] for example.

$$DRNN : \{\boldsymbol{a}; \boldsymbol{y}_{t-1}; \boldsymbol{C}_{t-1}\} \rightarrow \{\boldsymbol{p}_t; \boldsymbol{C}_t\} \tag{5}$$

sequence unrolled thus far and image regions attended to thus far (Equation 5). It is as complex model, comprised of the following sub-models (Figure 2b): 1) A LSTM-Stack [?] responsible for memorizing $\boldsymbol{C}_t$ and producing a recurrent activation $\boldsymbol{H}_t$, 2) A Visual attention and alignment model responsible for selecting relevant regions of the encoded image for input to the LSTM-Stack, [6] 3) A Deep Output Layer [?] that produces the output probabilities $\boldsymbol{p}_t$, 4) Init Model: A model that generates the initial state $\boldsymbol{C}_0$ and 5) An embedding matrix $\boldsymbol{E}$ (learned by training) that transforms $\boldsymbol{y}_t$ into a dense representation $\in \mathbb{R}^m$.

2.2.1 Inferencing

After the model is trained, the output sequence is generated by starting with the word 'bos' and then repeatedly sampling from $\boldsymbol{p}_t$ until <eos> is produced. The sequence of words thus sampled is the predicted sequence: $\hat{\boldsymbol{y}} := (\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_{\hat{\tau}})$; $\hat{\boldsymbol{y}}_t \in \mathbb{R}^K$. For this procedure we use beam search decoding [?] with a beam width of 10. Figure 1 shows an example predicted sequence and Figures 5 and 6 show examples of predictions rendered into images by a LaTeX 2_ε compiler.

2.2.2 Visual attention and alignment model

As previously alluded, the decoder soft selects/filters relevant (encoded) image regions at each step. This is implemented via. a 'soft attention' mechanism[7] which computes a weighted sum $\boldsymbol{z}_t$ of the pooled feature vectors $\boldsymbol{a}_l$. The visual attention model f_{att},

$$\boldsymbol{\alpha}_t := (\alpha_{t,1}, \ldots, \alpha_{t,L}) \,\Big|\, \begin{matrix} 0 \leq \alpha_{t,l} \leq 1 \\ \sum_l^L \alpha_{t,l} = 1 \end{matrix} \tag{6}$$

$$\boldsymbol{\alpha}_t = f_{att}(\boldsymbol{a}; \boldsymbol{H}_{t-1}) \tag{7}$$

$$\boldsymbol{z}_t = \boldsymbol{\alpha}_t \boldsymbol{a}^\top \tag{8}$$

computes the weight distribution $\boldsymbol{\alpha}_t$ (Equations 6, 7 and 8). f_{att} is modeled by an MLP (details in the appendix). While it is a possible for $\boldsymbol{\alpha}_t$ to end up uniformly distributed over $(\boldsymbol{a}_1 \ldots \boldsymbol{a}_L)$, in practice we see a unimodal shape with most of the weight concentrated on 1-4 neighborhood (see Figure 3) around the mode. We call this neighborhood the *focal-region* - i.e. the focus of attention. In other words we empirically observe that the attention model's focus is 'sharp'; converging towards the 'hard attention' formulation described by ?]. Also note that (Figure 3), the attention model is able to utilize the extra granularity available to it in the I2L-NOPOOL case and consequently generates much sharper focal-regions than I2L-STRIPS.

Furthermore, the model aligns the focal-region with the output word and thus scans text on the image left-to-right (I2L-STRIPS) or left-right and up-down (I2L-NOPOOL) just like a person would read it (Figure 3). We also observe that it doesn't focus on empty margins of the image except at the first and last (<eos>) steps which is quite intuitive for determining the beginning or end of text.

2.2.3 LSTM stack

The core sequence generator of the DRNN is a multilayer LSTM [?] (Figure 2b). Our LSTM cell implementation follows ?]. The LSTM cells are stacked in a multi-layer configuration [? ?] as in Equation 9. $LSTM^q$ is the LSTM cell at position q with $\boldsymbol{x}_t^q$, $\boldsymbol{h}_t^q$ and $\boldsymbol{c}_t^q$ being its input, hidden activation and cell state respectively. $LSTM^1$ receives the

$$LSTM^q \,:\, \{\boldsymbol{x}_t^q; \boldsymbol{h}_{t-1}^q; \boldsymbol{c}_{t-1}^q\} \rightarrow \{\boldsymbol{h}_t^q; \boldsymbol{c}_t^q\}$$
$$1 \leq q \leq Q \,;\, \boldsymbol{h}_t^q, \boldsymbol{c}_t^q \in \mathbb{R}^n$$
$$\boldsymbol{x}_t^q = \boldsymbol{h}_t^{q-1} \quad ; q \neq 1 \tag{9}$$
$$\boldsymbol{x}_t^1 = \{\boldsymbol{z}_t; \boldsymbol{E}\boldsymbol{y}_{t-1}\}$$

stack's input: soft attention context $\boldsymbol{z}_t$ and previous output word $\boldsymbol{E}\boldsymbol{y}_{t-1}$. $LSTM^Q$ produces the stack's output $\boldsymbol{H}_t = \boldsymbol{h}_t^Q$, which is sent up to the Deep Output Layer. Accordingly, the stack's activation ($\boldsymbol{H}_t$) and state ($\boldsymbol{C}_t$) are defined as: $\boldsymbol{H}_t = \boldsymbol{h}_t^Q$ and $\boldsymbol{C}_t := (\boldsymbol{c}_t^1, \ldots, \boldsymbol{c}_t^Q, \boldsymbol{h}_t^1, \ldots, \boldsymbol{h}_t^Q)$. We do not use skip or residual connections between the cells. Both of our models have two LSTM layers with $n = 1500$. Further discussion and details of this model can be found in the appendix.

---

[6]The LSTM-Stack and Visual Attention and Alignment model jointly form a Conditioned Attentive LSTM (CALSTM); $\boldsymbol{H}_t$ and $\boldsymbol{C}_t$ being its activation and internal state respectively. Our source-code implements the CALSTM as a RNN cell which may be used as a drop-in replacement for a RNN cell.

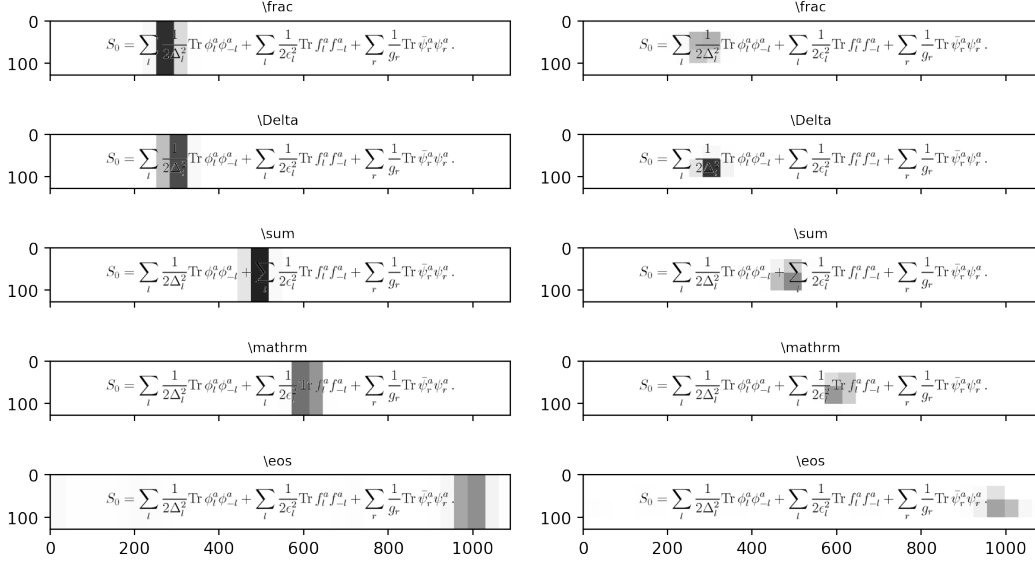[7]'Soft' attention as defined by ?] and originally proposed by ?].

Figure 3: Focal-regions learnt by the attention model: to the left by I2L-STRIPS and to the right by I2L-NOPOOL. Image darkness is proportional to $\boldsymbol{\alpha}_t$. Notice how $\boldsymbol{\alpha}_t$ concentrates on the image region corresponding to the output word (shown above the image). The \frac command starts a fraction, \mathrm sets a font and \eos is the <eos> token.

2.2.4 Deep output layer

We use a Deep Output Layer [?] to produce the final output probabilities: $\boldsymbol{p}_t = f_{out}(\boldsymbol{H}_t; \boldsymbol{z}_t; \boldsymbol{E}\boldsymbol{y}_{t-1})$. f_{out} is modeled by an MLP. Note that the output layer receives skip connections from the LSTM-Stack input (Equation 9). Details of this model can be found in the appendix.

2.2.5 Init model

The Init Model f_{init}, produces the initial state $\boldsymbol{C}_0$ of the LSTM-Stack. f_{init} is intended to 'look' at the entire image ($\boldsymbol{a}$) and setup the decoder appropriately before it starts generating the output.

$$f_{init} : \boldsymbol{a} \to (\boldsymbol{c}_0^1, \ldots, \boldsymbol{c}_0^Q, \boldsymbol{h}_0^1, \ldots, \boldsymbol{h}_0^Q) \quad (10)$$
$$\boldsymbol{h}_0^q, \boldsymbol{c}_0^q \in \mathbb{R}^n$$

That said, since it only provides a very small improvement in performance in exchange for over 7 million parameters, its need could be questioned. f_{init} is modeled as an MLP with common hidden layers and $2Q$ distinct output layers, one for each element of $\boldsymbol{C}_0$, connected as in Figure 4. See the appendix for more detail and discussion.
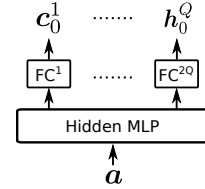

Figure 4: Init Model. FC = Fully Connected Layer.

2.3 Training

The entire model was trained end-to-end by minimizing the objective function $\mathcal{J}$ (Equation 11) using back propagation through time. The first term in Equation 11 is the average (per-word) log perplexity of the predicted sequence[8] and is the main objective. $\mathcal{R}$ is the

$$\mathcal{J} = -\frac{1}{\tau} log\left(P_r\left(\boldsymbol{y}|\boldsymbol{a}\right)\right) + \lambda_R \mathcal{R} \quad (11)$$

$$\mathcal{R} = \frac{1}{2}\sum_\theta \theta^2 \quad (11a)$$

L2-regularization term, equal to L2-norm of the model's parameters θ (weights and biases) and λ_R is a hyperparameter requiring tuning. Following ?] at first, we had included a penalty term intended to bias the distribution of the cumulative attention placed on an image-location $\alpha_l := \sum_{t=1}^\tau \alpha_{t,l}$. However we removed it for various reasons which are discussed in the appendix along with other details and analyses.

---

[8]i.e. Average cross-entropy, negative log-likelihood or negative log-probability.

We split the dataset into two fixed parts: 1) training dataset = 90-95% of the data and 2) test dataset 5-10%. At the beginning of each run, 5% of the training dataset was randomly held out as the validation-set and the remainder was used for training. Therefore, each such run had a different training/validation data-split, thus naturally cross-validating our learnings across the duration of the project. We trained the model in minibatches of 56 using the ADAM optimizer [**?**]; periodically evaluating it over the validation set[9]. For efficiency we batched the data such that each minibatch had similar length samples. For the final evaluation however, we fixed the training and validation dataset split and retrained our models for about 100 epochs ($\sim 2\frac{1}{2}$ days). We then picked the model-snapshots with the best validation BLEU score and evaluated the model over the test-dataset for publication. Table 1 lists the training parameters and metrics of various configurations. Training sequence predictions ($\hat{y}$) were obtained by CTC-decoding [**?**] p_t. Training BLEU score was then calculated over 100 consecutive mini-batches. We used two Nvidia GeForce 1080Ti graphics cards in a parallel towers configuration. Our implementation uses the Tensorflow toolkit and is distributed under AGPL license.

Table 1: Training metrics. $\lambda_R = 0.00005$ and $\beta_2 = 0.9$ for all runs. The number after @ sign is the training epoch of the selected model-snapshot. * denotes that the row corresponds to Table 2.

Dataset	Model	Init Model?	β_1	Train Epochs	Train BLEU	Validation BLEU	Valid'n ED
I2L-140K	I2L-STRIPS	Yes	0.5	104	0.9361	0.8900@72*	0.0677
	I2L-STRIPS	No	0.5	75	0.9300	0.8874@62	0.0691
	I2L-NOPOOL	Yes	0.5	104	0.9333	0.8909@72*	0.0684
	I2L-NOPOOL	No	0.1	119	0.9348	0.8820@92	0.0738
Im2latex-90k	I2L-STRIPS	Yes	0.5	110	0.9366	0.8886@77*	0.0688
	I2L-STRIPS	No	0.5	161	0.9386	0.8810@118	0.0750

3 Results

Given that there are multiple possible LATEXsequences that will render the same math image, ideally we should perform a visual evaluation. However, since there is no widely accepted visual evaluation metric, we report corpus BLEU (1,2,3 & 4 grams) and per-word Levenstein Edit Distance[10] scores (see Table 2). We also report a (non-standard) exact visual match score[103] which reports the percentage of exact visual matches, discarding all partial matches. While the predicted and targeted images match in at least 70%[103] of the cases, the model generates different but correct sequences (i.e. $y \neq \hat{y}$) in about 40% of the cases (Figure 5). For the cases where the images do not exactly match, the differences in most cases are minor (Figure 6). Overall, our models produce syntactically correct sequences[11] for at least 99.85% of the test samples (Table 2). Please visit our website to see hundreds of sample visualizations, analyses and discussions, data-set and source-code.

Table 2: Test results. Im2latex-100k results are from **?**]. The last column is the percentage of successfully rendering predictions.

Dataset	Model	BLEU Score	Edit Distance	Visual Match[103]	Compiling Predictions
I2L-140K	I2L-NOPOOL	**89.0**%	0.0676	70.37%	99.94%
	I2L-STRIPS	**89.0**%	0.0671	69.24%	99.85%
Im2latex-90k	I2L-STRIPS	**88.19**%	0.0725	68.03%	99.81%
Im2latex-100k	IM2TEX	87.73%	-	**79.88**%	-

---

[9]Evaluation cycle was run once or twice per epoch and/or when a training BLEU score calculated on sequences decoded using CTC-Decoding[**?**] jumped significantly.

[10]i.e. Edit distance divided by number of words in the target sequence.

[11]i.e. Those that were successfully rendered by LATEX 2_ε.

[103]We use the 'match without whitespace' algorithm provided by **?**] wherein two images count as matched if they match pixel-wise discarding white columns and allowing for upto 5 pixel image translation (a pdflatex quirk). It outputs a binary match/no-match verdict for each sample - i.e. partial matches however close, are considered a non-match.

	Input Image / Rendered Sequence	y_{len}	$\hat{y}_{len}$
0	$T^{+q}_{+2-2}=\frac{1}{2}\gamma^i_{q\dot{p}}\left(\Omega^{+2i}_{+2}\psi^{1-}_{-2\dot{p}}-\Omega^{+2i}_{-2}\psi^{1-}_{+2\dot{p}}\right),\ T^{+q}_{+p\pm2}=\frac{1}{2}\gamma^i_{q\dot{p}}\Omega^{+2i}_{+p}\psi^{1-}_{\pm2\dot{p}},$	147	155
1	$\sigma_{ij}(x^-,y^-;x^+)=\int\frac{dP^-}{4\pi}\frac{dp^+}{4\pi}\frac{dk^+}{4\pi}e^{-\frac{i}{2}P^-x^+}e^{-\frac{i}{2}(p^+x^--k^+y^-)}\sigma_{ij}(p^+,k^+;P^-),$	150	151
2	$G(f)^{(n)}_\beta=\Sigma^n_{m=0}\{\tilde{\Theta}^{(n-m)}_\beta,\tilde{f}^{(m)}\}_{(q)}+\Sigma^{(n-2)}_{m=0}\{\tilde{\Theta}^{(n-m)}_\beta,\tilde{f}^{(m+2)}\}_{(\phi)}+\{\tilde{\Theta}^{(n+1)}_\beta,\tilde{f}^{(1)}\}_{(\phi)}$	150	150
3	$S_0=\sum_l\frac{1}{2\Delta^2_l}\operatorname{Tr}\phi^a_l\phi^a_{-l}+\sum_l\frac{1}{2\epsilon^2_l}\operatorname{Tr}f^a_lf^a_{-l}+\sum_r\frac{1}{g_r}\operatorname{Tr}\bar{\psi}^a_r\psi^a_r\ .$	145	148
4	$ds^2=-\frac{t^2}{(t^2+r^2_-)(t^2-r^2_+)}dt^2+t^2(d\phi+\frac{r_+r_-}{t^2}dr)^2+\frac{(t^2+r^2_-)(t^2-r^2_+)}{t^2}dr^2.$	147	147
5	$H=\frac{1}{2E}U\begin{pmatrix}0&0&0\\0&\Delta m^2_{21}&0\\0&0&\Delta m^2_{31}\end{pmatrix}U^\dagger+\frac{1}{2E}\begin{pmatrix}a&\eta b&0\\\eta^*b&\eta'b&0\\0&0&0\end{pmatrix},$	147	147
6	$D^{ab}_{\mu\nu}(p,p3)=\frac{\delta^{ab}\delta^{a3}}{p^2-p^2_3+i\epsilon}\left[-g_{\mu\nu}+p_\mu p_\nu\left((1-\delta_{p3,0})\frac{1}{p^2_3}+\delta_{p3,0}(1-\xi)\frac{1}{p^2+i\epsilon}\right)\right]$	139	145
7	$V(H_1,H_2)=\frac{1}{8}(g^2_2+g^2_1)\left(|H_1|^2-|H_2|^2\right)^2+m^2_1|H_1|^2+m^2_2|H_2|^2-m^2_3(H_1H_2+\text{h.c.})$	144	145
8	$A^3_0(\alpha'\to0)=2g_d\ \varepsilon^{(1)}_\lambda\varepsilon^{(2)}_\mu\varepsilon^{(3)}_\nu\left\{\eta^{\lambda\mu}(p^\nu_1-p^\nu_2)+\eta^{\lambda\nu}(p^\mu_3-p^\mu_1)+\eta^{\mu\nu}(p^\lambda_2-p^\lambda_3)\right\}.$	146	145
9	$U^\dagger_LM_lU^l_R=M^*_l\ ,\ U^\dagger_LM_LU^*_L=M^*_L\ ,\ U^\dagger_LM_DU^\nu_R=M^*_D\ ,\ U^{\nu T}_RM_RU^\nu_R=M^*_R\ .$	149	145
10	$\sqrt{-g}g^{\mu_1\nu_1}g^{\mu_2\nu_2}...g^{\mu d-p}\nu^{d-p}\tilde{F}_{\nu_1\nu_2...\nu_{d-p}}=\frac{1}{p!}\epsilon^{\mu_1\mu_2...\mu_{d-p}\nu_1\nu_2...\nu_p}F_{\nu_1\nu_2...\nu_p},$	147	145
11	$\frac{dE}{dz}=\frac{dE_{el}}{dz}+\frac{dE_{rad}}{dz}\approx\frac{C_2\alpha_s}{\pi}\mu^2\ln\frac{3ET}{2\mu^2}\left(\ln\frac{9E}{\pi^3T}+\frac{3\pi^2\alpha_s}{2\mu^2}T^2\right).$	130	144
12	$L_0=(2n+1)\frac{|h|}{2}+|h|d^\dagger_0d_0-\frac{|h|}{2}-|h|\sum^\infty_{k=1}\left(d^\dagger_kd_k-\tilde{d}^\dagger_k\tilde{d}_k+a^\dagger_ka_k-b^\dagger_kb_k\right)+L^{free}_0.$	149	144
13	$Q_{7\gamma}=\frac{e}{8\pi^2}m_b\bar{q}_\alpha\sigma^{\mu\nu}(1+\gamma_5)b_\alpha F_{\mu\nu},\ \ Q_{8G}=\frac{g}{8\pi^2}m_b\bar{q}_\alpha\sigma^{\mu\nu}t^a_{\alpha\beta}b_\beta G^a_{\mu\nu},\ \ (q=d\text{ or }s).$	141	143
14	$ds^2=\alpha'\left(\frac{u^2h(u)}{R^2}\ e^{\gamma A}dx^2_0+\frac{u^2}{R^2}\ e^{\gamma C}dx^2_i+\frac{R^2}{u^2h(u)}\ e^{\gamma B}du^2+R^2\ e^{\gamma D}d\Omega^2_5\right),$	143	143
15	$\sin\left(\frac{\tilde{p}_1\cdot k}{2}\right)\sin\left(\frac{\tilde{p}_2\cdot k}{2}\right)\sin\left(\frac{\tilde{p}_3\cdot k}{2}\right)=-\frac{1}{4}(\sin\tilde{p}_1\cdot k+\sin\tilde{p}_2\cdot k+\sin\tilde{p}_3\cdot k)$	133	143
16	$[\mathcal{P}_0,X_0]=i,\ \ \ [\mathcal{P}_i,X_j]=-i\,\delta_{ij}\left(1-\frac{\tilde{\mathcal{P}}^2}{\kappa^2}\right)e^{\mathcal{P}_0/\kappa},\ \ \ [\mathcal{P}_0,X_i]=-\frac{2i}{\kappa}\mathcal{P}_ie^{\mathcal{P}_0/\kappa}$	139	143
17	$\langle J^{a_1}_{\mu_1}(P_1)...J^{a_n}_{\mu_n}(P_n)\rangle_T\approx(-i)^n\frac{NT^2}{12}\delta\Gamma^{a_1...a_n}_{\mu_1...\mu_n}(P_1,...,P_n)+O(\frac{1}{f^2_\pi})\ ,$	143	143
18	$\tilde{u}_\pi(\vec{k}'_1)^\dagger\tilde{u}_\pi(\vec{k}_1)=N^2_\pi\left[\cos^2\chi(\vec{k})-\frac{|\vec{p}|^2}{4}\left(\cos\chi(\vec{k})c_2(\vec{k})+\frac{1}{3}\vec{k}^2b_1(\vec{k})^2\right)\right].$	142	142
19	$\tau_0(y)=\sum^1_{\sigma_1=0}\cdot\cdot\sum^1_{\sigma_4=0}Y^{(0)}_{\sigma_1...\sigma_4}(t_1Z_1)^{\sigma_1}(t_2Z_2)^{\sigma_2}(t_5Z_5)^{\sigma_3}(t_8Z_8)^{\sigma_4}$	143	142
20	$\frac{\sin(2\beta)}{32\,\pi^2}I(\tilde{\Omega})\to\frac{\sin(2\beta)}{32\,\pi^2}(I(\tilde{\Omega})+c^2_2c^2_3\delta_1I(M_G,M_{G1})+c^2_2s^2_3\delta_2I(M_G,M_{G2}))$	142	142
21	$\mathcal{W}=Y_eL^jE^cH^i_1\epsilon_{ij}+Y_dQ^{ja}D^c_aH^i_1\epsilon_{ij}+Y_uQ^{ja}U^c_aH^i_2\epsilon_{ij}+\mu H^i_1H^j_2\epsilon_{ij}$	141	141
22	$-i\bar{\kappa}_{(\alpha)}\gamma^\mu\kappa_{(\beta)}\partial_\mu=-i\tilde{\mathcal{C}}_{\alpha\gamma}\Gamma_s(T^I)^\gamma_{\ \beta}\Gamma_{\text{Adj}}(u^{-1})^a_{\ I}e_a=-i\tilde{\mathcal{C}}_{\alpha\gamma}\Gamma_s(T^I)^\gamma_{\ \beta}k_{(I)}\ ,$	145	141

Figure 5: A sample of correct predictions by I2L-STRIPS. We've shown the long predictions hence lengths are touching 150. Note that at times the target length is greater than the predicted length and at times the reverse is true (though the original and predicted images were identical). All such cases would evaluate to a less than perfect BLEU score or edit-distance. This happens in about 40% of the cases. For more examples visit our website.

	y	$\hat{y}$
0	$\Psi:\tilde{S}^2\to\{\mathcal{M}_1,\mathcal{M}_2\}$	$\Psi:\tilde{S}^2\to\{\mathcal{M}_1,\mathcal{M}_2\}$
1	$\ln\frac{E+\sqrt{E^2-m^2_l}-m_\pi}{E-\sqrt{E^2-m^2_l}-m_\pi}.$	$\ln\frac{E+\sqrt{E^2-m^2_t}-m_\pi}{E-\sqrt{E^2-m^2_l}-m_\pi}.$
2	$\left(\frac{r_0}{\ell_s}\right)^{\tilde{d}}\sim g^{2-k}_s\ .$	$\left(\frac{r_0}{\ell_s}\right)^{\tilde{d}}\sim g^{2-k}_s\ .$
3	$\dot{p}_a+\epsilon^b_ap_b\omega_\mu\dot{x}I\mu=0$	$\dot{p}_a+\epsilon^b_ap_b\omega_\mu\dot{x}\,\mu_\mu=0$
5	$\hat{T}=\hat{V}+\hat{t}\hat{G}\hat{V},$	$\hat{T}=\hat{V}+\hat{f}\hat{G}\hat{V},$
7	$\Theta^A=(\vartheta^\alpha,\tilde{\vartheta}_{\dot{\alpha}}),\ \ \ \partial_A=(\partial_\alpha,\tilde{\partial}^{\dot{\alpha}}),\ \ \ \{\partial_A,\Theta^B\}=\delta^B_A$	$\Theta^A=(\vartheta^\alpha,\tilde{v}_{\dot{\alpha}}),\ \ \ \partial_A=(\partial_\alpha,\tilde{\partial}^{\dot{\alpha}}),\ \ \ \{\partial_A,\Theta^B\}=\delta^B_A$
8	$\hat{Y}_\tau(M_Z)|_{\overline{DR}}=\frac{m^{pole}_\tau-\Re e\Sigma_\tau(m^{pole}_\tau)|_{\overline{DR}}}{\hat{v}(M_Z)|_{\overline{DR}}\cos\beta(M_Z)}$	$\hat{Y}_\tau(M_Z)|_{\overline{DG}}=\frac{m^{vot}_r-kez_\tau(m^{pele}_r)|_{D\overline{H}}}{\delta(M_Z)|\overline{D}Ecs\beta(M_Z)}$
9	$3.4\beta^{[2|2]}(y)=-9y^2\left[\frac{1-22.21y+36.93y^2}{1-28.21y+143.2y^2}\right]$	$3.4\beta^{[2|2]}(y)=-9y^2\left[\frac{1-22.2y+36.93y^2}{1-28.2y+143.2y^2}\right]$
10	$\hat{D}^\alpha_{\ \beta}=\left(\Pi^2+2eBS_3\right)^\alpha_{\ \beta}$	$\hat{D}^\alpha_{\ \beta}=\left(\Pi^2+2eBS_3\right)^\alpha_{\ \beta}$
12	$\int\frac{d\bar{z}dzd\bar{b}db}{n}\epsilon V(f)=0$	$\int\frac{d\bar{z}dzd\bar{b}d}{n}\epsilon V(f)=0$
13	$\mathcal{L}^{(p^4)}_{\Delta S=1}=G_8F^2\sum^{37}_{i=1}N_iW_i$	$\mathcal{L}^{(s^0)}_{\Delta S^2=1}=G_8F^2\sum^{37}_{i=1}N_iW_i$
14	$\det(\mathcal{M}^{(0)}(N_0))=\alpha^{Q(0)}\prod_{r,s}[\alpha(h-h_{r,s})]^{P_\ell(N_0-rs/K)}\ ,$	$\det(\mathcal{M}^{(0)}(N_0))=\alpha^Q(0)\prod_{r,s}[\alpha(h-h_{r,s})]^{P_f(N_0-rs/K)}\ ,$
17	$\gamma\ \equiv\ \frac{e^2\,N}{2T}\ .$	$\gamma\ \equiv\ \frac{e^2\,N}{2T}\ .$

Figure 6: A random sample of mistakes made by I2L-STRIPS. Observe that usually the model gets most of the formula right and the mistake is only in a small portion of the overall formula (e.g. sample # 1; generating one subscript $_t$ instead of an $_l$). In some cases the mistake is in the font and in some cases the images are identical but were incorrectly flagged by the image-match evaluation software (e.g. sample # 0 & #17). In some cases the predicted formula appears more correct than the original! (sample # 10 where position of the subscript $_\beta$ has been 'corrected' by I2L-STRIPS).

3.1 Model Interpretability via attention

Since the LSTM stack only sees a filtered view (i.e. focal-region) of the input, it can only base its predictions on the focal-regions seen thus far and initial-state C_0. Further since the init-model has a negligible impact on performance we can drop it from the model (Table 1) and thereby the dependency on C_0 (now randomly initialized). Therefore if $\boldsymbol{I}_t$ is the focal-region at step t defined by the predicate $\boldsymbol{\alpha}_{t,l} > 0$, then $\boldsymbol{p}_t(\hat{\boldsymbol{y}}_t) = f_L(\boldsymbol{I}_t, \boldsymbol{I}_{t-1} \ldots \boldsymbol{I}_0)$ where f_L represents the LSTM-stack and Deep Output Layer. This fact aids considerably in interpreting the predictions of the model. We found heat-map type visuals of the focal-regions (Figure 3) very useful in interpreting the model even as we were developing it.

Object detection via attention: Additionally, we observe that the model settles on a step-by-step alignment of $\boldsymbol{I}_t$ with the output-word's location on the image: i.e. $\boldsymbol{p}_t(\hat{\boldsymbol{y}}_t) \approx f_L(\boldsymbol{I}_t)$. In other words $\boldsymbol{I}_t$ marks the bounding-box of $\hat{\boldsymbol{y}}_t$ even though we trained without any bounding-box data. Therefore our model -whose encoder has a narrow receptive field- can be applied to the object detection task without requiring bounding box training data, bottom-up region proposals or pretrained classifiers. Note that this is not possible with encoder architectures having wide receptive fields, e.g. those that employ a RNN [? ?] because their receptive fields encompass the entire input. A future work will quantify the accuracy of object detection [?] using more granular receptive fields. ?] have also used attention for object detection but their model is more complex in that it specifically models bounding-boxes although it doesn't require them for training.

3.2 Dataset

Datasets were created from single-line LaTeX math formulas extracted from scientific papers and subsequently processed as follows: 1) Normalize the formulas to minimize spurious ambiguity.[12] 2) Render the normalized formulas using pdflatex and discard ones that didn't compile or render successfully. 3) Remove duplicates. 4) Remove formulas with low-frequencey words (frequency-threshold = 24 for Im2latex-90k and 50 for I2l-140K). 5) Remove images bigger than 1086×126 and formulas longer than 150. Processing the Im2latex-100k dataset[104] (103559 samples) as above resulted in the **Im2latex-90k** dataset which has 93741 samples. Of these, 4648 were set aside as the test dataset and the remaining 89093 were split into training (95%) and validation (5%) sets before each run (section 2.3). We found the Im2latex-90k dataset too small for good generalization and therefore augmented it with additional samples from KDD Cup 2003. This resulted in the **I2L-140K** dataset with 114406 (training), 14280 (validation) and 14280 (test) samples. Since the normalized formulas are already space separated token sequences, no additional tokenization step was necessary. The vocabulary was therefore produced by simply identifying the set of unique space-separated words in the dataset.

Ancillary material All ancillary material: Both datasets, our model and data-processing source code, visualizations, result samples etc. is available at our website. Appendix is provided alongside this paper.

References

A   Qualitative analyses and details

This section is an appendix to the paper. We present here further details, analyses and discussion of our experiments and comparison with related work.

A.1   Encoder

Table 3 shows the configuration of the Encoder CNN. All convolution kernels have shape (3,3), stride (1,1) and $tanh$ non-linearity, whereas all maxpooling windows have shape (2,2) and stride (2,2). We initially experimented with the output of the VGG16 model [?] - per ?]. However (presumably since

---

[12]Normalization was performed using the method and software used by [?] which parses the formulas into an AST and then converts them back to normalized sequences.

[104]Im2latex-100k dataset is provided by [?].

	y	$\hat{y}$
0	$\frac{\partial A_{0\mu}}{\partial t}=-i[A_{0\mu},H_{F0}],$	$\frac{\partial A_{0\mu}}{\partial t}=-i[A_{0\mu},H_{F0}],$
1	$\{\Phi^i(x),\Phi^j(y)\}=\epsilon^{ij}\delta^2(x-y).$	$\{\Phi^i(x),\Phi^j(y)\}=\epsilon^{ij}\delta^2(x-y).$
2	$V_{total}=\sum_i\left|\frac{\partial W}{\partial z_i}\right|^2+V_D+V_{soft}$	$V_{total}=\sum_i\left|\frac{\partial W}{\partial z_i}\right|^2+V_D+V_{soft}$
3	$\alpha_\lambda^{\dagger a}(p)=\int d^3x\ e^{-ip\cdot x}\big[e^\lambda\cdot(\omega A^a-iE^a)+\int_\Omega(f_1\Pi^a+f_2\phi^a)\big]$	$\alpha_\lambda^{\dagger\alpha}(p)=\int d^3x\ e^{-ip\cdot x}\big[e^\lambda\cdot(\omega A^a-iE^a)+\int_\Omega(f_1\Pi^a+f_2\phi^a)\big]$
4	$H_{stat}(k)=P+\frac{i}{vk+i\epsilon},$	$H_{stat}(k)=P+\frac{i}{vk+i\epsilon},$
5	$(\phi^*P_s+\phi P_s^*)\frac{2\Delta^2}{M^2},$	$(\phi^*P_s+\phi P_s^*)\frac{2\Delta^2}{M^2},$
6	$H_{G/H}=\frac{1}{2}\left(\pi_\alpha-\frac{i\hbar}{2}\Gamma_\alpha\right)g^{\alpha\beta}\left(\pi_\beta+\frac{i\hbar}{2}\Gamma_\beta\right)=\frac{1}{2}\pi_\alpha g^{\alpha\beta}\pi_\beta+V_{G/H},$	$H_{G/H}=\frac{1}{2}\left(\pi_\alpha-\frac{i\hbar}{2}\Gamma_\alpha\right)g^{\alpha\beta}\left(\pi_\beta+\frac{i\hbar}{2}\Gamma_\beta\right)=\frac{1}{2}\pi_\alpha g^{\alpha\beta}\pi_\beta+V_{G/H},$
7	$S[\Phi]=S[\phi]+S[\varphi]+S_{\text{int}}[\phi,\varphi]$	$S[\Phi]=S[\phi]+S[\varphi]+S_{\text{int}}[\phi,\varphi]$
8	$\gamma_1=\frac{\kappa}{4\pi}\qquad\gamma_2=\frac{\lambda}{4}$	$\gamma_1=\frac{\kappa}{4\pi}$
9	$\Gamma=\frac{1-r^2}{8\pi M_B}(|M^S|^2+|M^P|^2).$	$\Gamma=\frac{1-r^2}{8\pi M_B}(|M^S|^2+|M^P|^2).$
10	$E(r)=-\left(\frac{2GE_\nu m_2}{r}\pm\frac{q_1q_2}{r}+\frac{m_\nu S_1S_2}{r}\right)$	$E(r)=-\left(\frac{2GE_\nu m_2}{r}\pm\frac{q_1q_2}{r}+\frac{m_\nu S_1S_2}{r}\right)$
11	$\chi(x_1,x_2)=\langle0|T\psi(x_1)\bar\psi(x_2)|P\rangle.$	$\chi(x_1,x_2)=\langle0|T\psi(x_1)\bar\psi(x_2)|P\rangle.$
12	$\epsilon L^{(2)}\theta=\bar{h}_1^{(2)}v^2\phi^j(\epsilon\gamma^i\theta)(\theta\gamma^{ij}\theta)+h_1^{(2)}v^iv^j\phi^k(\epsilon\gamma^i\theta)(\theta\gamma^{jk}\theta),$	$\epsilon L^{(2)}\theta=\bar{h}_1^{(2)}v^2\phi^j(\epsilon\gamma^i\theta)(\theta\gamma^{ij}\theta)+h_1^{(2)}v^iv^j\phi^k(\epsilon\gamma^i\theta)(\theta\gamma^{jk}\theta),$
13	$\sin^2 2\vartheta=\sin^2 2\vartheta_{\text{sun}}=\frac{4|U_{e1}|^2|U_{e2}|^2}{(|U_{e1}|^2+|U_{e2}|^2)^2}.$	$\sin^2 2\vartheta=\sin^2 2\vartheta_{\text{sun}}=\frac{4|U_{e1}|^2|U_{e2}|^2}{(|U_{e1}|^2+|U_{e2}|^2)^2}.$
14	$F_1=\frac{g^2}{192\pi^{5/2}}M_{Pl}\simeq1.5\times10^{15}$ GeV.	$F_1=\frac{g^2}{192\pi^{5/2}}M_{Pl}\simeq1.5\times10^{15}$ GeV.
15	$b_{<i>}=\prod_{0\le p<q\le p_i}X_{\tau^p(i)\tau^q(i)}(z,z).$	$b_{<i>}=\sum_{0\le p<q\le p}X_{r(i)\ i\tau(i)}(z,z).$
16	$F_1^{W+D}(x)=[d^p(x)+\bar{u}^p(x)+d^n(x)+\bar{u}^n(x)+2s(x)+2\bar{c}(x)]/2.$	$F_1^{W+D}(x)=[d^p(x)+\bar{u}^p(x)+d^n(x)+\bar{u}^n(x)+2s(x)+2\bar{c}(x)]/2.$
17	$u=q^2b^2/2\ \ or\ \ Q=Q_0u,\qquad Q_0=\frac{1}{Am_Nb^2}$	$u=q^2b^2/2\ \ or\ \ Q=Q_0u,\qquad Q_0=\frac{1}{Am_Nb^2}$
18	$\bar{h}_v\,\Gamma\,h_v=\frac{1}{2}\,\text{Tr}\left(\Gamma\,P_v\right)\bar{h}_v\,h_v-\frac{1}{2}\,\text{Tr}\left(\gamma_\mu\gamma_5\,P_v\,\Gamma\,P_v\right)\bar{h}_v\,\gamma^\mu\gamma_5\,h_v,$	$\bar{h}_v\,\Gamma\,h_v=\frac{1}{2}\,\text{Tr}(\Gamma\,P_v)\bar{h}_v\,h_v-\frac{1}{2}\,\text{Tr}(\gamma_\mu\gamma_5\,P_v\,\Gamma\,P_v)\bar{h}_v\,\gamma^\mu\gamma_5\,h_v,$
19	$\tilde{P}_g(z)=\Delta_{ns}\frac{1}{z}+\frac{1}{1-z}.$	$\tilde{P}_g(z)=\Delta_n\frac{1}{z}+\frac{1}{1-z}.$
20	$S\le S_{\text{H}},\ \ T\ge T_{\text{H}},\ \ E_c\le E_{\text{BH}},\ \ \text{for}\ HR\ge1$	$S\le S_{\text{H}},\ \ T\ge T_{\text{H}},\ \ E_c\le E_{\text{BH}},\ \ \text{for}\ HR\ge1$
21	$\Gamma\sim N^{-1/2}10^{23}\text{s}^{-1}\exp\left[-\frac{8\sqrt{2}}{3\cdot137}\left(\frac{-E}{m_e}\right)^{3/2}\frac{B_0}{NB}A^{1/2}\left(\frac{m_p}{m_e}\right)^{1/2}\right],$	$\Gamma\sim N^{-1/2}10^{23}\text{e}^{-1}\exp\left[-\frac{8\sqrt{2}}{3\cdot137}\left(\frac{-E}{m_e}\right)^{3/2}\frac{B_0}{NB}A^{1/2}\left(\frac{m_p}{m_e}\right)^{1/2}\right],$
22	$\frac{2|J|^2}{m_0{}^2}\cong\frac{m_b{}^2}{m_d{}^2}\sim2.5\times10^5.$	$\frac{2\int|^2}{m_0{}^2}\cong\frac{m_b^2}{m_d^2}\sim2.5\times10^5.$
23	$u=\frac{z}{\ell}U^{-1/2}\frac{\partial}{\partial t},$	$u=\frac{z}{\ell}U^{-1/2}\frac{\partial}{\partial t},$
24	$\Omega=\frac{\rho}{\rho_c}$	$\Omega=\frac{\rho}{\rho_c}$
25	$e^{(2r+1)\pi iL(0)}Y_1(v,x)e^{-(2r+1)\pi iL(0)}=Y_1((-1)^{L(0)}v,-x),$	$e^{(2r+1)\pi iL(0)}Y_1(v,x)e^{-(2r+1)\pi iL(0)}=Y_1((-1)^{L(0)}v,-x),$
26	$\mathbf{A}_2=\int d^2x\,\mathcal{A}_2(x)*\delta\alpha(x),$	$\mathbf{A}_2=\int d^2x\,\mathcal{A}_2(x)*\delta\alpha(x),$
27	$ds^2=(k+f_0\frac{R_0^2}{R^2})^{-1}dR^2+R^2d\Omega_k^2-(k+f_0\frac{R_0^2}{R^2})[dx^5+A_R(R)dR]^2$	$ds^2=(k+f_0\frac{R_0^2}{R^2})^{-1}dR^2+R^2d\Omega_k^2-(k+f_0\frac{R_0^2}{R^2})[dx^5+A_R(R)dR]^2$
28	$[\hat\rho_0,\hat\rho_0]=0,\quad[\hat{S}_0^A,$	$[\hat\rho_0,\hat\rho_0]=0,\quad[\hat{S}_0^A,$
29	$\mathcal{L}_4=(F^1-\partial_5A^2)\frac{dW}{dA^1}+\cdots.$	$\mathcal{L}_4=(F^1-\partial_5A^2)\frac{dW}{dA^1}+\cdots.$
30	$\Psi_j\overline{\Psi}_i=\delta_{ij}-q^{-1}\hat{\mathcal{R}}_{ikjl}\overline{\Psi}_l\Psi_k$	$\Psi_j\overline{\Psi}_i=\delta_{ij}-q^{-1}\hat{\mathcal{R}}_{iklj}\overline{\Psi}_l\Psi_k$
31	$\Pi_i=0,\qquad\Theta_k=\partial_k\Pi_0+cm^2h_k=0,$	$\Pi_i=0,\qquad\Theta_k=\partial_k\Pi_0+cm^2h_k=0,$
32	$\sin\delta=\frac{s_{23}c_{23}}{s_2c_2}\sin\delta_{13}$	$\sin\delta=\frac{s_{23}c_{23}}{s_2c_2}\sin\delta_{13}$
33	$\Delta_A=-2H\,s_{AA},$	$\Delta_A=-2H\,s_{AA},$
34	$\hat{D}_{\mu\nu}^{-1}=\hat{D}_{\mu\alpha}^{-1}\eta^{\alpha\beta}\left(\eta_{\beta\nu}+\sum_{n=1}^\infty A_n(D^{-1})_{\beta\nu}^n\right),$	$\hat{D}_{\mu\nu}^{-1}=\hat{D}_{\mu\alpha}^{-1}\eta^{\alpha\beta}\left(\eta_{\beta\nu}+\sum_{n=1}^\infty A_n(D^{-1})_{\beta\nu}^n\right),$
35	$H_G(x^2)=-\frac{1}{8\pi Ge^2}\left([B(x^2)]^{-2}-1\right),$	$H_G(x^2)=-\frac{1}{8\pi Ge^2}\left([B(x^2)]^{-2}-1\right),$
36	$T_{\mu\nu}=T_{\mu\nu}^++\frac{a_2^2}{a_1^2}T_{\mu\nu}^-.$	$T_{\mu\nu}=T_{\mu\nu}^++\frac{a_2^2}{a_1^2}T_{\mu\nu}^-.$
37	$G_{H^{d+1}}=\frac{l^{1-d}}{\Sigma_d}\int_{\frac{\sigma}{l}}^{+\infty}\frac{dx}{\sinh^d x}.$	$G_{H^{d+1}}=\frac{l^{1-d}}{\Sigma_d}\int_q^{+\infty}\frac{dx}{\sinh^d x}.$
38	$\Gamma_{\{\mu\}}^{(n)}=\frac{\delta^{(n)}\Gamma(A')}{\delta A'_{\mu_1}(x_1)\cdots\delta A'_{\mu_j}(x_j)\cdots\delta A'_{\mu_n}(x_n)},$	$\Gamma_{\{\mu\}}^{(n)}=\frac{\partial^{n(n)}\Gamma(A')}{\delta A'_{\mu_1}(x_1)\cdots\delta A'_{\mu_\mu}(x_j)\cdots\delta A_{\mu_n}^o(x_n)},$
39	$\dot\pi_{\text{ab}}=$	$\dot\pi_{\text{ab}}=$
40	$\gamma\pi_a^0=0,\ \gamma\pi_a^i=f^b_{ac}\pi_b^i\eta_2^c,\ \gamma\pi_{0i}=0,\ \gamma\pi_{ij}=0,$	$\gamma\pi_a^0=0,\ \gamma\pi_a^i=f^b_{ac}\pi_b^i\eta_2^c,\ \gamma\pi_{0i}=0,\ \gamma\pi_{ij}=0,$
41	$|Z_1|^2=|Z_2|^2=\frac{1}{(4G)^2}e^{-\eta_0}[(Q_R{}^1)^2+(Q_R{}^2)^2],$	$|Z_1|^2=|Z_2|^2=\frac{1}{(4G)^2}e^{-\eta_0}[(Q_R{}^1)^2+(Q_R{}^2)^2],$

Figure 7: A random sample of predictions of I2L-STRIPS containing both good and bad predictions. Note that though this is a random sample, prediction mistakes are not obvious and it takes some effort to point them out! For more examples visit our website.[105]

Table 3: Specification of the Encoder CNN.

Layer	Output Shape	Channels
Input (Image)	128×1088	1
Convolution	128×1088	64
Maxpool	64×544	64
Convolution)	64×544	128
Maxpool	32×272	128
Convolution	32×272	256
Maxpool	16×136	256
Convolution	16×136	512
Maxpool	8×68	512
Convolution	8×68	512
Maxpool	$4 \times 34 = (\acute{H} \times \acute{W})$	$512 = (\acute{D})$

VGG16 was trained on a different dataset and a different problem) the BLEU score didn't improve beyond 40%. Then we started training VGG16 along with our model but the end-to-end model didn't even start learning (the log-loss curve was flat) - possibly due to the large overall depth of the end-to-end model. Reducing the number of convolution layers to 6 and changing the non-linearity to $tanh$ (to keep the activations in check) got us good results. Further reducing number of layers to 5 yielded the same performance, therefore we stuck with that configuration (Table 3). In additon, we experimented with I2L-STRIPS because it reduces the rectangular image-map to a linear map, thereby presumably making the alignment model's task easier because now it would only need to scan in one-dimension. However, it performed around the same as I2L-NOPOOL and therefore that hypothesis was debunked. In fact we prefer I2L-NOPOOL since it has fewer parameters and its attention model has sharper focal-regions which helps with model interpretation.

A.2 Attention model

Table 4 specifies the configuration of the attention model MLP. **?** 's formulation of attention model ($\alpha_{t,l} = MLP(\boldsymbol{a}_l; \boldsymbol{H}_{t-1})$) receives inputs from only a single image location. In comparison, our formulation ($\boldsymbol{\alpha}_t = f_{att}(\boldsymbol{a}; \boldsymbol{H}_{t-1})$) receives the full encoded image $\boldsymbol{a}$ in its input. This change was needed because the previous formulation did not progress beyond a point, presumably because this problem warranted a wider receptive field. The new formulation works equally well with different pooling strides (and correspondingly different values of L).

Table 4: Specification of the Visual Attention Model MLP. L = 34 for I2L-STRIPS and and 136 for I2L-NOPOOL.

Layer	Num Units	Activation
3 (output)	L	softmax
2	max(128, L)	tanh
1	max(256, L)	tanh

Also, **?** 's formulation of $\boldsymbol{z}_t = \beta_t \cdot \boldsymbol{\alpha}_t \cdot \boldsymbol{a}$ includes a scalar $\beta_t = MLP(\boldsymbol{H}_{t-1})$ which informs the LSTM how much emphasis to place on the image v/s the language model. Experimentally we found that it had no impact on end-to-end performance, therefore we dropped it from our model.

? also use a simpler formula for $\mathcal{A} = \sum_{l=1}^{L} \left(\sum_{t=1}^{\tau} \alpha_{t,l} - 1 \right)^2$ which they call 'doubly stochastic optimization'. Our formulation uses the true mean of α_l, τ/L instead of 1, normalizes it to a fixed range so that it can be compared across models and more importantly, includes a target-ASE term ASE_T. Without this term, i.e. with $ASE_T = 0$, $\mathcal{A}$ would bias the attention model towards uniformly scanning all the L image locations. This is undesirable since there are many empty regions of the images where it makes no sense for the attention model to spend much time. Conversely, there are some densely populated regions (e.g. a symbol with complex superscript and subscripts) where the model would reasonably spend more time because it would have to produce a longer output sequence. In other words, the optimal scanning pattern would have to be non-uniform - $ASE_T \neq 0$. Also, the scanning pattern would vary from sample to sample, but ASE_T is set to a single value (even if zero) for all samples. Therefore we preferred to remove the attention-model bias altogether from the objective function by setting $\lambda_A = 0$ in all situations except when the attention model needed a 'nudge' in order to 'get off the ground'. In such cases we set ASE_T based on observed values of ASE_N (Table 8).
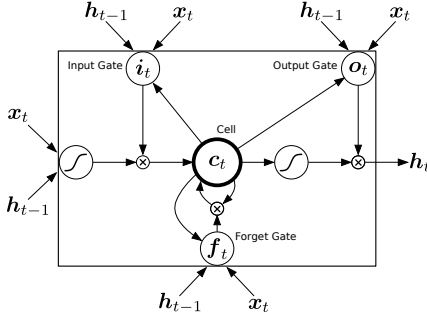
A.3 LSTM stack

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma\left(W_{xi}\boldsymbol{x}_t + W_{hi}\boldsymbol{h}_{t-1} + W_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right) \\
\boldsymbol{f}_t &= \sigma\left(W_{xf}\boldsymbol{x}_t + W_{hf}\boldsymbol{h}_{t-1} + W_{cf}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t\tanh\left(W_{xc}\boldsymbol{x}_t + W_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right) \\
\boldsymbol{o}_t &= \sigma\left(W_{xo}\boldsymbol{x}_t + W_{ho}\boldsymbol{h}_{t-1} + W_{co}\boldsymbol{c}_t + \boldsymbol{b}_o\right) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t\tanh(\boldsymbol{c}_t) \\
&\quad \boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t, \boldsymbol{c}_t, \boldsymbol{h}_t \in \mathbb{R}^n
\end{aligned} \tag{12}$$

Figure 8: LSTM Cell

Our LSTM cell implementation (Figure. 8 and equation 12) follows **? ?**]. In equation 12 σ is the logistic sigmoid function and $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$, $\boldsymbol{c}_t$ and $\boldsymbol{h}_t$ are respectively the *input gate*, *forget gate*, *output gate*, *cell* and *hidden* activation vectors of size n.

During experimentation our penultimate LSTM-stack which had 3 LSTM layers with 1000 units each, gave us a validation score of 87.45%. At that point experimental observations suggested that the LSTM stack was the accuracy 'bottleneck' because other sub-models were performing very well. Increasing the number of LSTM units to 1500 got us better validation score - but a worse overfit. Reducing the number of layers down to 2 got us the best overall validation score. In comparison, **?**] have used a single LSTM layer with 1000 cells.

A.4 Deep output layer

Note that the output layer receives skip connections from the LSTM-Stack input ($\boldsymbol{p}_t = f_{out}(\boldsymbol{H}_t; \boldsymbol{z}_t; \boldsymbol{E}\boldsymbol{y}_{t-1})$). We observed a 2% impact on the BLEU score with the addition of input-to-output skip-connections. This leads us to believe that adding skip-connections within the LSTM-stack may help further improve model accuracy. Overall accuracy also improved by increasing the number of layers from 2 to 3. Lastly, observe that this sub-model is different

Table 5: Configuration of the Deep Output Layer MLP. $K = 339$ and 358 for I2L-140K and Im2latex-90k datasets respectively.

Layer	Num Units	Activation
3 (output)	K	softmax
2	max(358, K)	tanh
1	max(358, K)	tanh

from **?**] wherein the three inputs are affine-transformed into D dimensions, summed and then passed through one fully-connected layer. After experimenting with their model we ultimately chose to instead feed the inputs (concatenated) to a fully-connected layer thereby allowing the MLP to naturally learn the input-to-output function. We also increased the number of layers to 3, changed activation function of hidden units from relu to tanh[101] and ensured that each layer had at least as many units as the softmax layer (K).

A.5 Init model

The init model MLP is specified in Table 6. We questioned the need for the Init Model and experimented just using zero values for the initial state. That caused a slight but consistent decline ($< 1\%$) in the validation score, indicating that the initial state learnt by our Initial State Model did contribute in some way towards learning and generalization. Note however that our Init

Table 6: Init Model layers.

Layer	Num	Units	Activation Function
Output	2Q	n	tanh
Hidden	1	100	tanh

Model is different than **?** , in that our version uses all L feature vectors of $\boldsymbol{a}$ while theirs takes the

---

[101] We changed from relu to tanh partly in order to remedy 'activation-explosions' which were causing floating-point overflow errors.

average. We also added a hidden layer and used $tanh$ activation function instead of $relu$. We did start off with their version but that did not provide an appreciable impact to the bottom line (validation). This made us hypothesize that perhaps taking an average of the feature vectors was causing a loss of information; and we mitigated that by taking in all the L feature vectors without summing them. After making all these changes, the Init Model yields a consistent albiet small performance improvement (Table. 7). But given that it consumes $\sim$7.5 million parameters, its usefulness remains in question.

Table 7: Impact of the Init Model on overall performance. Since it comprises 10-12% of the total params, it may as well be omitted in exchange for a small performance hit.

Model	Init Model Present?	Validation BLEU	Num Params
I2L-NOPOOL	Yes	89.09%	7,569,300
I2L-NOPOOL	No	88.20%	0
I2L-STRIPS	Yes	89.00%	7,569,300
I2L-STRIPS	No	88.74%	0

A.6 Training and dataset

A.6.1 Alpha penalty

Please see equations 13 through 13e. The loss function equation stated in the paper is Equation 13 but with λ_A set to 0. That was the case when training models who's results we have published, however at other times we had included a penalty term $\lambda_A \mathcal{A}$ which we discuss next. Observe that while $\sum_l^L \alpha_{t,l} = 1$, there is no constraint on how the attention is distributed across the L locations of the image. The term $\lambda_A \mathcal{A}$ serves to steer the variance of α_l by penalizing any deviation from a desired value. ASE (Alpha Squared Error) is the sum of squared-difference between α_l and its mean τ/L; and ASE_N is its normalized value [13] $\in [0,100]$[14]. Therefore $ASE_N \propto ASE \propto \sigma_{\alpha_l}^2$. ASE_T which is the desired value of ASE_N, is a hyperparameter that needs to be discovered through experimentation[15]. Table 8 shows training results with alpha-penalty details.

Table 8: Training metrics. $\lambda_R = 0.00005$ and $\beta_2 = 0.9$ for all runs.

Dataset	Model	Init Model?	λ_A	β_1	Training Epochs	Training BLEU	Validation ED	$\overline{ASE_N}$
I2L-140K	I2L-STRIPS	Yes	0.0	0.5	104	0.9361	0.0677	5.3827
	I2L-STRIPS	No	0.0	0.5	75	0.9300	0.0691	4.9899
	I2L-NOPOOL	Yes	0.0	0.5	104	0.9333	0.0684	4.5801
	I2L-NOPOOL	No	0.0	0.1	119	0.9348	0.0738	4.7099
Im2latex-90k	I2L-STRIPS	Yes	0.0	0.5	110	0.9366	0.0688	5.1237
	I2L-STRIPS	No	0.0005	0.5	161	0.9386	0.0750	4.8291

Default values of β_1 and β_2 of the ADAM optimizer - 0.9 and 0.99 - yielded very choppy validation score curves with frequent down-spikes where the validation score would fall to very low levels, ultimately resulting in lower peak scores. Reducing the first and second moments (i.e. β_1 and β_2) fixed the problem suggesting that the default momentum was too high for our 'terrain'. We did not use dropout for regularization, however increasing the data-set size (I2L-140K) and raising the minimum-word-frequency

$$\mathcal{J} = -\frac{1}{\tau} log\left(P_r\left(\boldsymbol{y}|\boldsymbol{a}\right)\right) + \lambda_R \mathcal{R} + \lambda_A \mathcal{A} \quad (13)$$

$$\mathcal{R} = \frac{1}{2}\sum_\theta \theta^2 \quad (13a)$$

$$\mathcal{A} = (ASE_N - ASE_T) \quad (13b)$$

$$ASE_N = \frac{100}{\tau^2\left(\frac{L-1}{L}\right)} \cdot ASE \quad (13c)$$

$$ASE = \sum_{l=1}^{L}\left(\alpha_l - \frac{\tau}{L}\right)^2 \quad (13d)$$

$$\sum_{t=1}^{?} \quad (13e)$$

[13]It can be shown that $\tau^2\left(\frac{L-1}{L}\right)$ is the maximum possible value of ASE.

[14]We normalize ASE so that it may be compared across batches, runs and models.

[15]Start with $ASE_T = 0$, observe where ASE_N settles after training, then set ASE_T to that value and repeat until approximate convergence.

threshold from 24 (Im2latex-90k) to 50 ((I2L-140K)) did yield better generalization and overall test scores (Table 8). Finally, normalizing the data[16] yielded about 25% more accuracy than without.

---

[16]Normalization was performed using the method and software used by [**?**] which parses the formulas into an AST and then converts them back to normalized sequences.