Teaching Machines to Code: Neural Markup Generation with Visual Attention

Sumeet Singh [1]

Abstract

We present a deep recurrent neural network model with 'soft visual attention' that learns to generate LaTeX markup of real-world math formulas from their images. Applying neural sequence generation techniques that have been very successful in the fields of machine translation and image/handwriting/speech captioning, recognition, transcription and synthesis; we create an image-to-markup model that learns to produce syntactically and semantically correct LaTeX markup code of over 150 words long and achieves a BLEU score of 89%; the best reported so far for the Im2Latex problem[100]. We also visually demonstrate that the model learns to scan the image left-right / up-down much as a human reads text.

1. Introduction

In the past decade, deep neural network models based on RNNs[1] and/or CNNs[2] have been shown to be very powerful sequence modelers. 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** (van den Oord et al., 2016b;c; Salimans et al., 2017; Theis & Bethge, 2015), **image description** (Karpathy & fei Li, 2015; Xu et al., 2015; Johnson et al., 2016; Pedersoli et al., 2016; Vinyals et al., 2015), **video description** (Donahue et al., 2015), **speech and audio synthesis** (van den Oord et al., 2016a), **handwriting recognition** (Graves & Schmidhuber, 2008; Bluche, 2016), **handwriting synthesis** (Graves, 2013), **machine translation** (Cho et al., 2014; Bahdanau et al., 2014; Kalchbrenner et al., 2016; Sutskever et al., 2014), **speech recognition** (Graves et al., 2006; Chan et al., 2015; Graves et al., 2013), **etc.** (Graves, 2008)

One class of sequence models employ the so-called *encoder-decoder* (Cho et al., 2014) or *sequence-to-sequence* (Sutskever et al., 2014) 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 (for e.g. in machine translation use-cases) or different modalities (for 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[3]. 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 (Bahdanau et al., 2014; Luong et al., 2015). Among other things, this architecture has been used for image-captioning.We employ a variant of this architecture to map images of math formulas into LaTeX markup code. The contributions of this paper are: 1) Solves the Im2Latex problem[100] and achieves the best reported BLEU score. 2) Pushes the limits of the neural encoder-decoder architecture with visual attention. 3) Analyses variations of the model and cost function. Specifically we note the changes needed to the base model (Xu et al., 2015) in order to take the BLEU score from 40% to 89%.

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 (see section 2.5) was produced by rendering single-line real-world LaTeX 2_ε 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 s, is comprised of an image $\boldsymbol{x}^{(s)}$ - the input - and a corresponding target markup-sequence $\boldsymbol{y}^{(s)}$ - the output

[1]Saratoga, California, USA. Correspondence to: Sumeet Singh <sumeet@singhonline.info>.

[100]https://openai.com/requests-for-research/#im2latex

[1]Recurrent Neural Network

[2]Convolutional Neural Networks and variants such as dilated CNNs (Yu & Koltun, 2015)

[3]generally backprop or reinforcement learning

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

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

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

- of length $\tau^{(s)}$. Each word of the sequence $\boldsymbol{y}_t^{(s)}$, belongs to the vocabulary of the dataset plus two special tokens, begin-of-sequence 'bos' and end-of-sequence 'eos' (Figure 1). Denoting image dimensions as H_I, W_I and C_I and the vocabulary as a set V of K words and dropping the superscript (s) for brevity, we represent:

$$\boldsymbol{x} := \mathrm{Array}(H_I, W_I, C_I) \tag{1a}$$
$$\boldsymbol{y} := \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_\tau\}; \quad \boldsymbol{y}_t \in V \tag{1b}$$
$$V := \{\text{dataset vocabulary}, \text{eos}, \text{bos}\}; |V| = K \tag{1c}$$

The task is to generated markup that a LaTeX 2_ε 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 2) has the same basic architecture as Xu et al. (2015) in the way the encoder, decoder and a visual attention interact. [4]

2.1. Encoder

IMAGE PREPROCESSING: All images are standardized to the same size by centering the text 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 rectangular *visual feature map* $\acute{\boldsymbol{A}}$, of $\acute{H} \times \acute{W}$ *visual feature vectors* $\boldsymbol{a}_{(\acute{h},\acute{w})}$ of size $\acute{D}$. $\acute{H}$ and $\acute{W}$ are the visual feature map's dimensions along the height and width axes of the image respectively. Table 1 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). See section A.2 for further comments around this architecture.

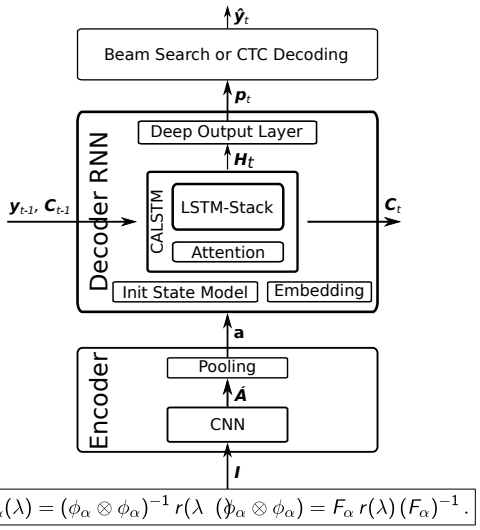

Figure 2. Model outline showing major parts of the model. Beam search decoder is only used during inferencing, not training. LSTM-Stack and Attention model jointly form a Conditioned Attentive LSTM stack (CALSTM) which can itself be stacked. 3 shows more details of the decoder.

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	$4 \times 34 = (\acute{H} \times \acute{W})$	$512 = (\acute{D})$

Table 1. Specification of the Encoder CNN.

Next, the visual feature vectors are pooled together in rectangular blocks of shape $[S_H, S_W]$; resulting in *pooled feature vectors* $\boldsymbol{a}_{(h,w)}$ of size $D = \acute{D} \cdot S_H \cdot S_W$. The shape of the overall feature map reduces to $[H, W]$ where $H = \acute{H}/S_H$

[4] However there are significant differences between the two models which we detail in the remainder of this paper and in section A.1.

Model	Pooling Stride	Pooled Map Shape	Num Parameters
I2L-NOPOOL	[1,1]	[4,34]	61,235,049
I2L-STRIPS	[4,1]	[1,34]	70,984,635

Table 2. Pooling configuration of two model variants.

and $W = S_W/\acute{W}$:

$$\boldsymbol{A} := \begin{Bmatrix} \boldsymbol{a}_{(1,1)} & \dots & \boldsymbol{a}_{(1,W)} \\ \vdots & \vdots & \vdots \\ \boldsymbol{a}_{(H,1)} & \dots & \boldsymbol{a}_{(H,W)} \end{Bmatrix}; \quad \boldsymbol{a}_{(h,w)} \in \mathbb{R}^D$$

By varying S_H and S_W one can vary the receptive field of the pooled feature vectors. We share results of two models, one with stride [1,1] (i.e. no pooling) and the other [4,1] (Table 2). Each pooled feature vector is a window into a distinct spatial region of the image; specifically the rectangle projected by its receptive field.[5] The idea is to partition the encoded image so as to provide the decoder (Section 2.2) with the opportunity to select only relevant regions of the image as needed at a given time-step t. Bahdanau et al. 2014 showed that such piecewise encoding enables modeling longer sequences as opposed to models that encode the entire input into a single feature vector (Sutskever et al., 2014; Cho et al., 2014). [6] We represent $\boldsymbol{A}$ as a flattened sequence $\boldsymbol{a}$ of pooled feature vectors $\boldsymbol{a}_l$ (Equation 2):

$$\boldsymbol{a} := \begin{Bmatrix} \boldsymbol{a}_1 & \dots & \boldsymbol{a}_L \end{Bmatrix}; \quad \boldsymbol{a}_l \in \mathbb{R}^D \qquad (2)$$
$$L{=}HW \; ; \; l{=}H(h{-}1){+}w$$

2.2. Decoder

The decoder is a Recurrent Neural Network (RNN) that models the discrete probably distribution of the output word $\boldsymbol{y}_t$, conditioned on the sequence of previous words $\boldsymbol{y}_{<t}$ and the encoded image $\boldsymbol{a}$[7]. Denoting the decoder's output by $\boldsymbol{p}_t$ we define:

$$\begin{aligned} \boldsymbol{p}_t &: \{1, \dots, K\} \rightarrow [0, 1] \\ \boldsymbol{y_t} &\sim \boldsymbol{p}_t \\ P_r(\boldsymbol{y}_t|\boldsymbol{y}_{<t}, \boldsymbol{a}) &= \boldsymbol{p}_t(\boldsymbol{y}_t) \end{aligned} \qquad (3)$$

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

[6]That said, Bahdanau et al. 2014 employ a bidirectional-LSTM (Graves, 2008) 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 Deng et al. 2017 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.

[7]This is now a very standard way to model sequence probabilities in neural sequence-generators. See (Sutskever et al., 2014) for example.

Probability of the output sequence for image $\boldsymbol{a}$ follows:

$$P_r(\boldsymbol{y}|\boldsymbol{a}) = \prod_{t=1}^{\tau} P_r(\boldsymbol{y}_t|\boldsymbol{y}_{<t}, \boldsymbol{a}) = \prod_{t=1}^{\tau} \boldsymbol{p}_t(\boldsymbol{y}_t) \qquad (4)$$

The decoder (Figure 3) is an LSTM (Hochreiter & Schmidhuber, 1997) based model with an internal state $\boldsymbol{C}_t$ that holds relevant information (features) regarding the output sequence unfolded thus far, the image regions attended to and an initial state generated at $t = 0$. In addition the decoder receives as inputs, the previous word $\boldsymbol{y}_{t-1}$ and encoded image $\boldsymbol{a}$. This architecture enables it to condition its output ($\boldsymbol{p}_t$) on the entire previous sequence and the image. Representing it as a function $f_{\mathcal{D}}$, we have:

$$f_{\mathcal{D}} : \{\boldsymbol{a};\, \boldsymbol{y}_{t-1};\, \boldsymbol{C}_{t-1}\} \rightarrow \{\boldsymbol{p}_t;\, \boldsymbol{C}_t\} \qquad (5)$$

It is comprised of the following sub-models (Figure 2):

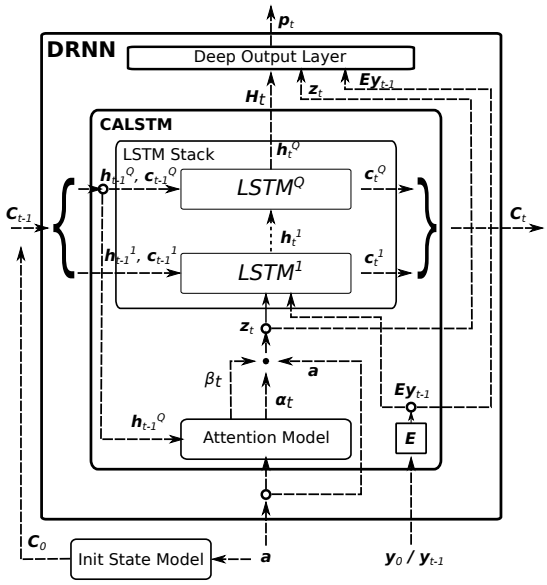

Figure 3. Schematic of Decoder RNN showing its sub-models. There are two nested RNN cells in all: The decoder RNN (DRNN) at the top level, nesting the CALSTM which nests the LSTM-Stack. The Init Model does not participate in recurrence, therefore its is shown outside the box.

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.

NOTE: 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.[8]

3) A Deep Output Layer over the LSTM-Stack (Pascanu et al., 2013) that produces the output probabilities $\boldsymbol{p}_t$.

4) Init Model: A model that generates the initial state $\boldsymbol{C}_0$.

5) An embedding matrix $\boldsymbol{E}$ (learnt through training) for transforming $\boldsymbol{y}_t$ 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 $\hat{\boldsymbol{y}}$ (Figure 2).

$$\hat{\boldsymbol{y}} := \{\hat{\boldsymbol{y}}_1, \ldots, \hat{\boldsymbol{y}}_{\hat{\tau}}\}\,; \qquad \hat{\boldsymbol{y}}_t \in \mathbb{R}^K \tag{6}$$

For this procedure - also called 'decoding'[9] - we use *beam search decoding* with a *beam width* of 10.

2.2.2. Visual Attention and Alignment Model

As previously alluded, the decoder selects relevant regions of the image at each step. This is implemented via. a 'soft attention' mechanism[10] which computes a weighted mean $\boldsymbol{z}_t$ (equation 9) of the pooled feature vectors $\boldsymbol{a}_l$. The visual attention model f_{att}, computes the weight distribution $\boldsymbol{\alpha}_t$.

$$\boldsymbol{\alpha}_t := \{\alpha_{t,1}, \ldots, \alpha_{t,L}\} \tag{7}$$
$$0 \leq \alpha_{t,l} \leq 1\,; \quad \sum_l^L \alpha_{t,l} = 1$$
$$\boldsymbol{\alpha}_t = f_{att}\left(\boldsymbol{a}; \boldsymbol{H}_{t-1}\right) \tag{8}$$
$$\boldsymbol{z}_t = \boldsymbol{\alpha}_t \cdot \boldsymbol{a} \tag{9}$$

f_{att} is modeled by an MLP (Table 3):

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

Table 3. Specification of the Visual Attention Model MLP. L = 34 for I2L-STRIPS and and 136 for I2L-NOPOOL.

Experiments: While there is a potential for $\boldsymbol{\alpha}_t$ to be uniformly distributed all across $\{\boldsymbol{a}_1 \ldots \boldsymbol{a}_L\}$, in practice we see that most of its mass gets concentrated on a 2-4 region neighborhood (see Figure 4). That is, the model learns to focus its attention on small *focal-regions* of the image, somewhat similar to the 'hard attention' formulation described by (Xu et al., 2015) except that in this case the focal-regions are learnt, not 'hard coded'. Also, notice that for the same example (Figure 4) the attention model is able to exploit the extra granularity of image-map available in the I2L-NOPOOL case and consequently generates a much smaller focal-region than I2L-STRIPS. It would be interesting to experiment with an even more granular image-map to see if the attention model refines its focal-region even further.

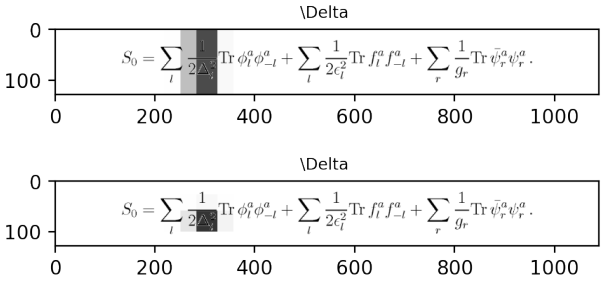

Figure 4. Focal-regions learnt by the attention model: at the top by I2L-STRIPS and at the bottom by I2L-NOPOOL. Image darkness is proportional to $\boldsymbol{\alpha}_t$. The focal-regions are 3-4 blocks big and surround the symbol (Δ) whose markup is being produced at that step. Also, the model is able to utilize the finer granularity of I2L-NOPOOL and produces a tighter focal region than I2L-STRIPS. More samples at our website.

Further, the model aligns its attention with the words it is producing and scans the image left-to-right (I2L-STRIPS) or left-right/up-down (I2L-NOPOOL) very similar to how a human would read a piece of text (Figure 5).

2.2.3. LSTM Stack

The core sequence-generator of the decoder is a multilayer LSTM (Hochreiter & Schmidhuber, 1997; Graves, 2013) (Figure 3). Our LSTM cell implementation[11] (Figure. 6) follows Graves et al. (2013):

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma\left(W_{xi}\boldsymbol{x}_t + W_{hi}\boldsymbol{h}_{t-1} + W_{ci}\boldsymbol{c}_{t-1} + \boldsymbol{b}_i\right)\\
\boldsymbol{f}_t &= \sigma\left(W_{xf}\boldsymbol{x}_t + W_{hf}\boldsymbol{h}_{t-1} + W_{cf}\boldsymbol{c}_{t-1} + \boldsymbol{b}_f\right)\\
\boldsymbol{c}_t &= \boldsymbol{f}_t\boldsymbol{c}_{t-1} + \boldsymbol{i}_t \tanh\left(W_{xc}\boldsymbol{x}_t + W_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c\right)\\
\boldsymbol{o}_t &= \sigma\left(W_{xo}\boldsymbol{x}_t + W_{ho}\boldsymbol{h}_{t-1} + W_{co}\boldsymbol{c}_t + \boldsymbol{b}_o\right)\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \tanh(\boldsymbol{c}_t)\\
&\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t, \boldsymbol{c}_t, \boldsymbol{h}_t \in \mathbb{R}^n
\end{aligned} \tag{10}$$

where 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*

[8] Our source-code implements the CALSTM as a RNN cell which may be used as a drop-in replacement for a RNN cell.

[9] The term 'decoding' is borrowed from Hidden Markov Models. For further details see for e.g. (Graves, 2008)

[10] 'soft' attention as defined by Xu et al. (2015) and originally proposed by (Bahdanau et al., 2014)

[11] We've used a Tensorflow implementation of LSTM

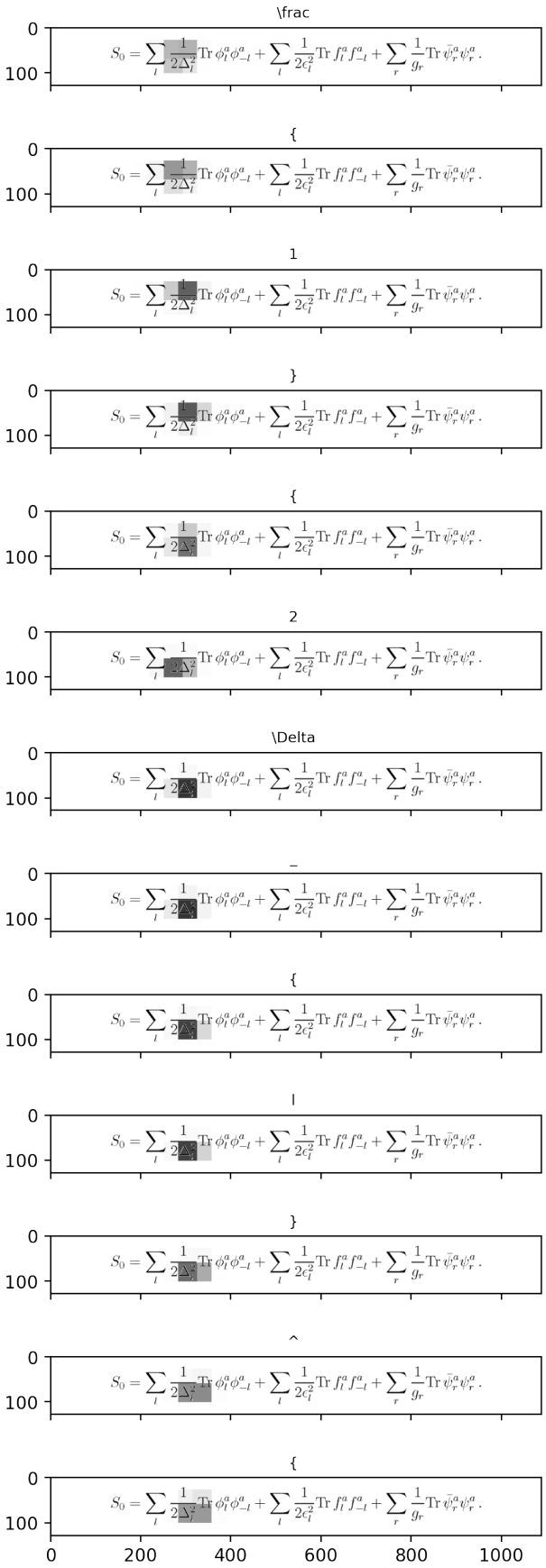

Figure 5. Attention scan at steps 12-24. The movement of focal-region is aligned with the output word (above the image). Continued in Figure 8.

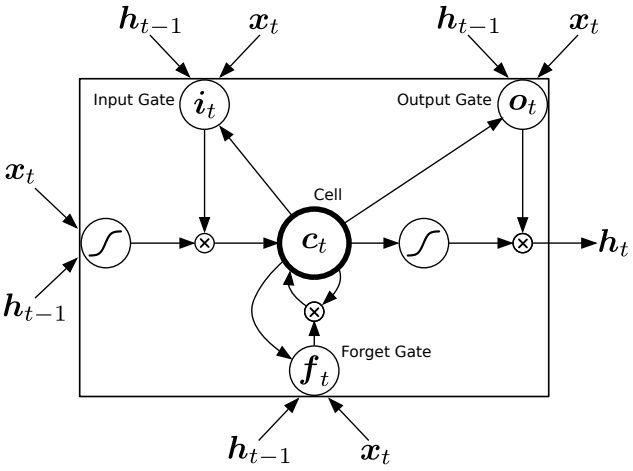

Figure 6. LSTM Cell

gate, *cell* and *hidden* activation vectors of size n. The LSTM cells are stacked in a multi-layer configuration (Zaremba et al., 2014; Pascanu et al., 2013) as below:

$$LSTM^q: \quad \{\boldsymbol{x}_t^q; \boldsymbol{h}_{t-1}^q; \boldsymbol{c}_{t-1}^q\} \rightarrow \{\boldsymbol{h}_t^q; \boldsymbol{c}_t^q\} \quad (11)$$
$$1 \leq q \leq Q; \quad \boldsymbol{h}_t^q, \boldsymbol{c}_t^q \in \mathbb{R}^n$$
$$\boldsymbol{x}_t^q = \boldsymbol{h}_t^{q-1} \quad ; q \neq 1 \quad (12)$$
$$\boldsymbol{x}_t^1 = \{\boldsymbol{z}_t; \boldsymbol{E}\boldsymbol{y}_{t-1}\} \quad (13)$$

$LSTM^q$ 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 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: its hidden activation $\boldsymbol{h}_t^Q$ which is sent up to the Deep Output Layer. We do not use skip or residual connections between the cells. Accordingly, activation $(\boldsymbol{H}_t)$ and state $(\boldsymbol{C}_t)$ of the LSTM-Stack[12] are defined as:

$$\boldsymbol{H}_t := \boldsymbol{h}_t^Q$$
$$\boldsymbol{C}_t := \{\boldsymbol{c}_t^1, \ldots, \boldsymbol{c}_t^Q, \boldsymbol{h}_t^1, \ldots, \boldsymbol{h}_t^Q\} \quad (14)$$

Both of our models have two LSTM layers with $n = 1500$. EXPERIMENTS: During experimentation our penultimate model 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. DIFFERENCES: In comparison, Xu et al. (2015) have used a single LSTM layer with 1000 cells.

[12] $\boldsymbol{H}_t$ and $\boldsymbol{C}_t$ are also activation and state of the CALSTM.

2.2.4. Deep Output Layer

We use a a Deep Output Layer (Pascanu et al., 2013) to produce the final output probabilities $\boldsymbol{p}_t$:

$$\boldsymbol{p}_t = f_{out}(\boldsymbol{H}_t;\, \boldsymbol{z}_t;\, \boldsymbol{E}\boldsymbol{y}_{t-1}) \tag{15}$$

f_{out} is modeled by an MLP (Table 4).

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

Table 4. Configuration of the Deep Output Layer MLP. K = 339 and 358 for I2L-140K and Im2latex-90k datasets respectively.

Experiments: Note that the output layer receives skip connections from the LSTM-Stack input (Equation 13). 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. Also, accuracy improved by increasing the number of layers from 2 to 3. Lastly, observe that this sub-model is different from Xu et al. (2015) wherein the three inputs are affine-transformed into a D dimensions, summed and then passed through one fully-connected layer. After experimenting with their model we ultimately chose to feed the inputs (concatenated) to a fully-connected layer thereby allowing the MLP to naturally learn the input-to-output function instead of forcing a summation. 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 (i.e. K).

2.2.5. Init Model

The Init Model f_{init}, produces the initial state C_0 of the LSTM-Stack, based on the pooled feature vectors:

$$f_{init} : \{\boldsymbol{a}\} \rightarrow \{\boldsymbol{c}_0^1, \ldots, \boldsymbol{c}_0^Q,\, \boldsymbol{h}_0^1, \ldots, \boldsymbol{h}_0^Q\} \tag{16}$$
$$\boldsymbol{h}_0^q, \boldsymbol{c}_0^q \in \mathbb{R}^n$$

f_{init} is modeled as an MLP with common hidden layers and $2Q$ distinct output layers, one for each element of C_0, connected as in Figure 7 and specified in Table 5. That said, since it only provides a very small improvement in performance in exchange for over 7 million parameters, its presence could be questioned (see Section A.3). The idea behind the init model is to setup the decoder to extract relevant features from the image (if any) before it starts generating the sequence.

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

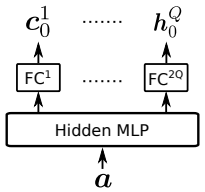

Figure 7. Init Model. FC = Fully Connected Layer.

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

Table 5. Init Model layers.

2.3. Training

The entire model was trained end-to-end by minimizing the objective function $\mathcal{J}$ (Equation 17) using back propagation through time :

$$\mathcal{J} = -\frac{1}{\tau} log\left(P_r\left(\boldsymbol{y}|\boldsymbol{a}\right)\right) + \lambda_R\mathcal{R} + \lambda_A\mathcal{A} \tag{17}$$
$$= -\frac{1}{\tau} log\left(P_r\left(\boldsymbol{y}|\boldsymbol{a}\right)\right) + \lambda_R\mathcal{R} \qquad ;\lambda_A = 0$$
$$\mathcal{R} = \frac{1}{2}\sum_{\theta}\theta^2 \tag{17a}$$
$$\mathcal{A} = (ASE_N - ASE_T) \tag{17b}$$
$$ASE_N = \frac{100}{\tau^2\left(\frac{L-1}{L}\right)} \cdot ASE \tag{17c}$$
$$ASE = \sum_{l=1}^{L}\left(\alpha_l - \frac{\tau}{L}\right)^2 \tag{17d}$$
$$\alpha_l := \sum_{t=1}^{\tau}\alpha_{t,l} \tag{17e}$$

The first term in equation 17 is the average (per-word) log perplexity of the predicted sequence[13] and is the main objective. $\mathcal{R}$ is the L2-regularization term, equal to L2-norm of the model's parameters θ (weights and biases). $\mathcal{A}$ is an optional penalty term intended to bias the distribution of α_l (Equation 17e) - the amount of attention placed on image-location l. 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

[13] Also variously known as cross-entropy, negative log-likelihood or negative log-probability

ASE_N is its normalized value [14] $\in$ [0,100][15]. Therefore $ASE_N \propto ASE \propto \sigma^2_{\alpha_l}$. ASE_T which is the desired value of ASE_N, is a hyperparameter that needs to be discovered through experimentation[16]. λ_R and λ_A are also hyperparameters that need to be tuned. See section A.1 for more discussion around this topic.

For experimentation 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 through the experimentation period (few months). We trained the model in minibatches of 56 using the ADAM optimizer (Kingma & Ba, 2014) for updating parameters; periodically evaluating it over the validation set[17]. 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 ($2\frac{1}{2}$ days on I2L-140K dataset). We then picked the model-snapshots with the best validation BLEU score and evaluated the model over the test-dataset for publication. Training as well as evaluation was executed on two Nvidia GeForce 1080Ti graphics cards using a parallel towers architecture. Our implementation[102] uses the Tensorflow toolkit and is distributed under AGPL license.

Dataset	Model	Training Epochs	Validation BLEU @Epoch
I2L-140K	I2L-STRIPS	104	0.8900@72
	I2L-NOPOOL	104	0.8909@72
Im2latex-90k	I2L-STRIPS	110	0.8886@77

Table 6. Training parameters. $\lambda_A = 0$, $\lambda_R = 0.00005$, $\alpha = 0.0001$, $\beta_1 = 0.5$ and $\beta_2 = 0.9$ for all runs. Init Model was present in all runs. See table 11 for more training data.

2.4. 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[18] scores (see table 7). We also report exact visual match score[103] which reports the percentage of exact visual match, discarding all partial matches. While the predicted and targeted images match[103] in around 70% of the cases, the model generates different but correct sequences (i.e. $\boldsymbol{y} \neq \hat{\boldsymbol{y}}$) in about 40% of the cases (Figure 8). For the 30% cases where the images do not match, the differences in most cases are minor (Figure 9). Overall, our models produce syntactically correct sequences[19] for at least 99.85% of the test samples (Figure 12).

2.5. Datasets

Datasets were created by processing single-line LaTeX math formulas extracted from scientific papers as follows: 1) Normalize the formulas to minimize spurious ambiguity.[20] 2) Render the normalized formulas using pdflatex and discard ones that didn't compile or render successfully, 3) Filter out duplicates, 4) Remove formulas with low-frequencey words (frequency-threshold = 24 for Im2latex-90k and =50 or I2l-140K), 5) Remove samples with images bigger than 1086×126, 6) Remove formulas greater than 150 words long. 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 as described in 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. The result is the **I2L-140K** dataset which has 114406, 14280 and 14280 training, validation and test samples respectively. Since the normalized formulas are already space separated token sequences, no additional tokenization step was necessary. The vocabulary was generated by simply identifying the set of unique space-separated words in the dataset. Both datasets are available at our website[105] and the processing code is available at our source-code repository[102].

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

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

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

[17] Evaluation cycle was run once or twice per epoch and/or when a training BLEU score calculated on sequences decoded using CTC-Decoding(Graves et al., 2006) jumped significantly.

[102] Source Code: https://github.com/untrix/im2latex

[18] i.e. edit distance divided by number of words in the target sequence.

[19] i.e. those that were successfully rendered by LaTeX 2_ε

[103] We use the 'match without whitespace' algorithm provided by Deng et al. (2017) 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.

[20] Normalization was performed using the method and software used by (Deng et al., 2017) which parses the formulas into an AST and then converts them back to normalized sequences.

[104] Im2latex-100k dataset is provided by (Deng et al., 2017)

[105] Paper's Website: https://untrix.github.io/im2latex/

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

Table 7. Test results. IM2TEX results are taken from Deng et al. (2017). Im2latex-90k dataset are the 90k samples of the Im2latex-100k dataset (Deng et al., 2017) that remained after the formulas were normalized, rendered and filtered of duplicates, non-compiling and large formulas.

	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	$S_0=\sum_l\frac{1}{2\Delta^2_l}\mathrm{Tr}\,\phi^a_l\phi^a_{-l}+\sum_l\frac{1}{2\epsilon^2_l}\mathrm{Tr}\,f^a_lf^a_{-l}+\sum_r\frac{1}{g_r}\mathrm{Tr}\,\bar{\psi}^a_r\psi^a_r$.	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	$D^{ab}_{\mu\nu}(p,p_3)=\frac{\delta^{ab}\delta^{a3}}{p^2-p^2_3+i\epsilon}\left[-g_{\mu\nu}+p_\mu p_\nu\left((1-\delta_{p_3,0})\frac{1}{p^2_3}+\delta_{p_3,0}(1-\xi)\frac{1}{p^2+i\epsilon}\right)\right]$	139	145
7	$V(H_1,H_2)=\frac{1}{8}(g^2_2+g^2_1)\left(\lvert H_1\rvert^2-\lvert H_2\rvert^2\right)^2+m^2_1\lvert H_1\rvert^2+m^2_2\lvert H_2\rvert^2-m^2_3(H_1H_2+\mathrm{h.c.})$	144	145
8	$A^3_0(\alpha'\to0)=2g_d\ \varepsilon^{(1)}_\lambda\varepsilon^{(2)}_\mu\varepsilon^{(3)}_\nu\left\{\eta^{\lambda\mu}(p^\nu_1-p^\nu_2)+\eta^{\lambda\nu}\left(p^\mu_3-p^\mu_1\right)+\eta^{\mu\nu}\left(p^\lambda_2-p^\lambda_3\right)\right\}$.	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	$\sqrt{-g}g^{\mu_1\nu_1}g^{\mu_2\nu_2}\dots g^{\mu_{d-p}\nu_{d-p}}\tilde{F}_{\nu_1\nu_2\dots\nu_{d-p}}=\frac{1}{p!}\epsilon^{\mu_1\mu_2\cdots\mu_{d-p}\nu_1\nu_2\cdots\nu_p}F_{\nu_1\nu_2\dots\nu_p}$,	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	$L_0=(2n+1)\frac{\lvert h\rvert}{2}+\lvert h\rvert d^\dagger_0d_0-\frac{\lvert h\rvert}{2}-\lvert h\rvert\sum^\infty_{k=1}\left(d^\dagger_kd_k-\tilde{d}^\dagger_k\tilde{d}_k+a^\dagger_ka_k-b^\dagger_kb_k\right)+L^{free}_0$.	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	$[\mathcal{P}_0,X_0]=i,\quad[\mathcal{P}_i,X_j]=-i\,\delta_{ij}\left(1-\frac{\vec{\mathcal{P}}^2}{\kappa^2}\right)e^{\mathcal{P}_0/\kappa},\quad[\mathcal{P}_0,X_i]=-\frac{2i}{\kappa}\,\mathcal{P}_i\,e^{\mathcal{P}_0/\kappa}$	139	143
17	$\langle J^{a_1}_{\mu_1}(P_1)\dots J^{a_n}_{\mu_n}(P_n)\rangle_T\approx(-i)^n\frac{NT^2}{12}\delta\Gamma^{a_1\cdots a_n}_{\mu_1\cdots\mu_n}(P_1,\dots,P_n)+O(\frac{1}{f^2_\pi})$,	143	143
18	$\tilde{u}_\pi(\vec{k}'_1)^\dagger\tilde{u}_\pi(\vec{k}_1)=N^2_\pi\left[cos^2\chi(\vec{k})-\frac{\lvert\vec{p}\rvert^2}{4}\left(cos\chi(\vec{k})c_2(\vec{k})+\frac{1}{3}\vec{k}^2b_1(\vec{k})^2\right)\right]$.	142	142
19	$\tau_0(y)=\sum^1_{\sigma_1=0}\cdot\cdot\sum^1_{\sigma_4=0}Y^{(0)}_{\sigma_1\cdots\sigma_4}(t_1Z_1)^{\sigma_1}(t_2Z_2)^{\sigma_2}(t_5Z_5)^{\sigma_3}(t_8Z_8)^{\sigma_4}$	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	$-i\bar{\kappa}_{(\alpha)}\gamma^\mu\kappa_{(\beta)}\partial_\mu=-i\tilde{\mathcal{C}}_{\alpha\gamma}\Gamma_s(T^I)^\gamma{}_\beta\Gamma_{\mathrm{Adj}}(u^{-1})^a{}_Ie_a=-i\tilde{\mathcal{C}}_{\alpha\gamma}\Gamma_s(T^I)^\gamma{}_\beta k_{(I)}$,	145	141
23	$\mathcal{L}_{\mathrm{eff}}=\sum_{i,j}\frac{1}{2}K_{\phi_i,\bar{\phi}_j}\nabla\phi_i\cdot\nabla\bar{\phi}_j-\mathrm{e}^K\left[\sum_{i,j}K^{\phi_i\ \bar{\phi}_j}(D_{\phi_i}W)(D_{\bar{\phi}_j}W)^\dagger-3\lvert W\rvert^2\right]$	150	141
24	$T^{(l)}=\int\frac{d^4p_n}{(2\pi)^4}\ \overline{u}_Y(p'_Y)\left\{\left(\sum^6_{j=1}\mathcal{A}_j\mathcal{M}_j\right)S(p_p)\Gamma_dCS(p_n)T_{YN}S(p_Y)\right\}\overline{u}(p'_N)$.	140	140
25	$\sigma^{(\pi^2)}_{q\bar{q}}(s,m^2)=\sigma^{(0)}_{q\bar{q}}(s,m^2)\left[\frac{\pi\alpha_s(\mu^2)}{2\beta}\left(C_F-\frac{C_A}{2}\right)+\frac{\alpha_s(\mu^2)}{\pi}C_F\left(\frac{9}{2}+\frac{\pi^2}{3}\right)\right]$.	140	140
26	$21\frac{d\bar{x}_2}{dt}+(2\beta_0\bar{x}_0(t))\bar{x}_2=-\beta_2\bar{x}^4_0(t)-3\beta_1\bar{x}^2_0(t)\bar{x}_1(t)-\beta_0\bar{x}^2_1(t),\quad\bar{x}_2(0)=0,$	140	139
27	$G^{ab}_{\mu\nu}(p)=\frac{-i\delta^{ab}}{(2\pi)^{2\omega}(p^2+i\epsilon)}\left[g_{\mu\nu}-\frac{p_\mu n_\nu+p_\nu n_\mu}{p\cdot n}+\frac{n^2p_\mu p_\nu}{(p\cdot n)^2}\right],\ n^2>0, epsilon>0,$	139	139
28	$\tilde{S}=\int d^2x\left\{g^2(1+\varphi^2_1)^2(\partial\tilde{\varphi}_0)^2+8\theta g^2\partial_a\tilde{\varphi}_0\partial^a\varphi_1+\frac{(1+16\theta^2g^4)}{g^2(1+\varphi^2_1)^2}(\partial\varphi_1)^2\right\}$	140	138
29	$p^2_1p^2_2-(p_1\cdot p_2)^2=-\frac{1}{4}(p^4_1+p^4_2+p^4_5-2p^2_1p^2_2-2p^2_1p^2_5-2p^2_2p^2_5)=0$	138	138
30	$g_{ij,kl}=\frac{-q^{-1}}{[2]}D_{si}\hat{R}^\epsilon_{js,kl}=\frac{-q^{-1}}{[2]}\epsilon_{im}\hat{R}_{jm,kt}\epsilon^{-1}_{lt}\quad,\quad(g=\frac{-q^{-1}}{[2]}D_1\mathcal{P}\hat{R}^\epsilon_{12})\quad,$	138	138

Table 8. 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.[105]

	$\boldsymbol{y}$	$\hat{\boldsymbol{y}}$
0	$\Psi{:}\tilde{S}^2{\to}\{\mathcal{M}_1,\mathcal{M}_2\}$	$\Psi{:}\tilde{S}^2{\to}\{\mathcal{M}_1,\mathcal{M}_2\}$
1	$\ln\frac{E+\sqrt{E^2-m_l^2}-m_\pi}{E-\sqrt{E^2-m_l^2}-m_\pi}.$	$\ln\frac{E+\sqrt{E^2-m_t^2}-m_\pi}{E-\sqrt{E^2-m_l^2}-m_\pi}.$
2	$\left(\frac{r_0}{\ell_s}\right)^{\bar{d}}{\sim}g_s^{2-k}.$	$\left(\frac{r_0}{\ell_s}\right)^{\bar{d}}{\sim}g_s^{2-k}.$
3	$\dot{p}_a{+}\epsilon_a^b p_b\omega_\mu\dot{x}I\mu{=}0$	$\dot{p}_a{+}\epsilon_a^b p_b\omega_\mu\dot{x}\,\mu_\mu{=}0$
4	$[M_{(m)}^{\alpha,\beta},M_{(n)}^{\alpha',\beta'}]{\equiv}M_{(m)}^{\alpha,\beta}M_{(n)}^{\alpha',\beta'}{-}(-)^{\alpha\beta'+\beta\alpha'}e^{2i(\beta'm-\beta n)x}M_{(n)}^{\alpha',\beta'}M_{(m)}^{\alpha,\beta}$	$[M_{(m)}^{\alpha,\beta},M_{(n)}^{\alpha',\beta'}]{\equiv}M_{(m)}^{\alpha,\beta}M_{(n)}^{\alpha',\beta'}{-}(-)^{\alpha\beta'+\beta\alpha'}e^{2i(\beta'm-\beta-n)x}M_{(n)}^{\alpha',\beta'}M_{(m)}^{\alpha,\beta}$
5	$\hat{T}{=}\hat{V}{+}\hat{t}\hat{G}\hat{V},$	$\hat{T}{=}\hat{V}{+}\hat{f}\hat{G}\hat{V},$
6	$\gamma^{(n)}{=}\gamma^{a_1\cdots a_n}{=}\gamma^{[a_1}\gamma^{a_2}...\gamma^{a_n]}{=}\frac{1}{n!}\sum_{perm'_s}(-1)^p\gamma^{a_1}...\gamma^{a_n}$	$\gamma^{(n)}{=}\gamma^{a_1\cdots a_n}{=}\gamma^{[a_1}\gamma^{a_2}...\gamma^{a_n]}{=}\frac{1}{n!}\sum_{perm_s}(-1)^p\gamma^{a_1}...\gamma^{a_p}$
7	$\Theta^A{=}\left(\vartheta^\alpha,\widetilde{\vartheta}_{\dot{\alpha}}\right),\quad \partial_A{=}\left(\partial_\alpha,\widetilde{\partial}^{\dot{\alpha}}\right),\quad \left\{\partial_A,\Theta^B\right\}{=}\delta_A^B$	$\Theta^A{=}(\vartheta^\alpha,\widetilde{v}_{\dot{\alpha}}),\quad \partial_A{=}\left(\partial_\alpha,\widetilde{\partial}^{\dot{\alpha}}\right),\quad \left\{\partial_A,\Theta^B\right\}{=}\delta_A^B$
8	$\hat{Y}_\tau(M_Z)\vert_{\overline{DR}}{=}\frac{m_\tau^{pole}-\Re e\Sigma_\tau(m_\tau^{pole})\vert_{\overline{DR}}}{\hat{v}(M_Z)\vert_{\overline{DR}}cos\beta(M_Z)}$	$\hat{Y}_\tau(M_Z)\vert_{\overline{DG}}{=}\frac{m_r^{vot}-\mathrm{kez}_\tau(m_r^{pele})\vert_{\overline{DH}}}{\delta(M_Z)\vert\overline{D}Ecs\beta(M_Z)}$
9	$3.4\beta^{[2\vert2]}(y){=}{-}9y^2\left[\frac{1-22.21y+36.93y^2}{1-28.21y+143.2y^2}\right]$	$3.4\beta^{[2\vert2]}(y){=}{-}9y^2\left[\frac{1-22.2y+36.93y^2}{1-28.2y+143.2y^2}\right]$
10	$\widehat{D}^\alpha{}_\beta{=}\left(\Pi^2{+}2eBS_3\right)^\alpha{}_\beta$	$\widehat{D}^\alpha{}_\beta{=}\left(\Pi^2{+}2eBS_3\right)^\alpha_\beta$
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	$\mathcal{L}^{(p^4)}_{\Delta S=1}{=}G_8F^2\sum_{i=1}^{37}N_iW_i$	$\mathcal{L}^{(s^0)}_{\Delta S^2=1}{=}G_8F^2\sum_{i=1}^{37}N_iW_i$
14	$\det(\mathcal{M}^{(0)}(N_0)){=}\alpha^{Q(0)}\prod_{r,s}[\alpha(h{-}h_{r,s})]^{P_\ell(N_0-rs/K)},$	$\det(\mathcal{M}^{(0)}(N_0)){=}\alpha^{Q}(0)\prod_{r,s}[\alpha(h{-}h_{r,s})]^{P_f(N_0-r_s/K)},$
15	$\beta_{++}{=}\frac{r^2}{4M^2}{-}M^2yv^2{+}\frac{1}{2}\left[\frac{M^2}{M^2}(1{-}y){-}1\right]rs_+{+}M^2\frac{\vec{p'}^2}{M^2}s_+^2,$	$\beta_{++}{=}\frac{r^2}{4M^2}{-}M^2y^2{+}\frac{1}{2}\left[\frac{M^2}{M^2}(1{-}y){-}1\right]r_{s+}{+}M^2\frac{p'}{M^2}s^2,$
16	$\mathbf{Q}_m^a(\zeta)X{\equiv}(S_m(\zeta),X)^a{-}i\hbar\bar{\Delta}_m^aX,\qquad \mathbf{Q}_A(\zeta)X{\equiv}\{S_m(\zeta),X\}_A{-}i\hbar\bar{\Delta}_AX$	$\mathbb{K}_m^a(\zeta)X{\equiv}(S_m(\zeta),X)^a{-}i\hbar\bar{\Delta}_m^aX,\qquad \mathbf{Q}_A(\zeta)X{\equiv}\{S_m(\zeta),X\}_A{-}i\hbar\bar{\Delta}_AX.$

Table 9. A random sample of mistakes made by I2L-STRIPS. Observe that the model gets the formula right in most part and makes a mistake in only a small part 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!(sample # 10 where position of the subscript $_\beta$ has been 'corrected' by I2L-STRIPS). For more examples visit our website.[105]

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

Bluche, Théodore. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *NIPS*, 2016.

Bluche, Théodore, Ney, Hermann, and Kermorvant, Christopher. A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition. In *SLSP*, 2014.

Chan, William, Jaitly, Navdeep, Le, Quoc V., and Vinyals, Oriol. Listen, attend and spell. *CoRR*, abs/1508.01211, 2015. URL http://arxiv.org/abs/1508.01211.

Cho, Kyunghyun, van Merrienboer, Bart, aglar Gülehre, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Deng, Yuntian, Kanervisto, Anssi, and Rush, Alexander M. What you get is what you see: A visual markup decompiler. *CoRR*, abs/1609.04938, 2016.

Deng, Yuntian, Kanervisto, Anssi, Ling, Jeffrey, and Rush, Alexander M. Image-to-markup generation with coarse-to-fine attention. In *ICML*, 2017.

Donahue, Jeff, Hendricks, Lisa Anne, Guadarrama, Sergio, Rohrbach, Marcus, Venugopalan, Subhashini, Saenko, Kate, and Darrell, Trevor. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389, 2015.

Graves, Alex. Supervised sequence labelling with recurrent neural networks. In *Studies in Computational Intelligence*, 2008.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Graves, Alex and Schmidhuber, Jürgen. Offline handwriting recognition with multidimensional recurrent neural networks. In *NIPS*, 2008.

Graves, Alex, Fernández, Santiago, Gomez, Faustino J., and Schmidhuber, Jürgen. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *ICML*, 2006.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey E. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013. URL http://arxiv.org/abs/1303.5778.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Johnson, Justin, Karpathy, Andrej, and Fei-Fei, Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2016.

Kalchbrenner, Nal, Espeholt, Lasse, Simonyan, Karen, van den Oord, Aäron, Graves, Alex, and Kavukcuoglu, Koray. Neural machine translation in linear time. *CoRR*, abs/1610.10099, 2016.

Karpathy, Andrej and fei Li, Fei. Deep visual-semantic alignments for generating image descriptions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3128–3137, 2015.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

Pascanu, Razvan, aglar Gülehre, Cho, Kyunghyun, and Bengio, Yoshua. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026, 2013.

Pedersoli, Marco, Lucas, Thomas, Schmid, Cordelia, and Verbeek, Jakob. Areas of attention for image captioning. *CoRR*, abs/1612.01033, 2016. URL http://arxiv.org/abs/1612.01033.

Salimans, Tim, Karpathy, Andrej, Chen, Xi, and Kingma, Diederik P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

Theis, Lucas and Bethge, Matthias. Generative image modeling using spatial lstms. In *NIPS*, 2015.

van den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W., and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016a.

van den Oord, Aäron, Kalchbrenner, Nal, and Kavukcuoglu, Koray. Pixel recurrent neural networks. In *ICML*, 2016b.

van den Oord, Aäron, Kalchbrenner, Nal, Vinyals, Oriol, Espeholt, Lasse, Graves, Alex, and Kavukcuoglu, Koray. Conditional image generation with pixelcnn decoders. *CoRR*, abs/1606.05328, 2016c.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Xu, Kelvin, Ba, Jimmy, Kiros, Jamie Ryan, Cho, Kyunghyun, Courville, Aaron C., Salakhutdinov, Ruslan, Zemel, Richard S., and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

A. Qualitative Analyses and Discussion

In this section we present further analyses and discussion of our experiments and comparison with related work.

A.1. Attention Model

Xu et al. 2015'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 (equation 8) 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).

Xu et al. 2015'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.

Xu et al. 2015 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, since the scanning pattern would vary from sample to sample a single value of ASE_T for all samples is also wrong. Therefore unless necessary, we should completely remove all attention-model bias from the objective function and allow it to learn freely. We went with that approach and set $\lambda_A = 0$ in all situations except where the attention model needed a 'nudge' in order to 'get off the ground'; when we set ASE_T to values observed in previous runs (Table 11).

A.2. Encoder

We initially experimented with the output of the VGG16 model (Simonyan & Zisserman, 2014) - per Xu et al. (2015). However (presumably since the model was trained on a different dataset and a different problem) the BLEU score didn't improve beyond 40%. Then we started training the CNN along with the model but that was even worse in that the 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 1). We chose to experiment with I2L-STRIPS because it reduces the rectangular image-map to a linear map, thereby presumably making the alignment model's task easier. However, it performed around the same as I2L-NOPOOL and therefore that hypothesis was debunked.

A.3. Init Model

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 Model is different than (Xu et al., 2015), in that our version uses all L feature vectors of $\boldsymbol{a}$ while theirs takes the average. We also added a hidden layer and used $tanh$ activation function instead of $relu$. We did start off with their version of the init model but that did not provide an appreciable impact to the bottom line (validation), which made us hypothesize that perhaps

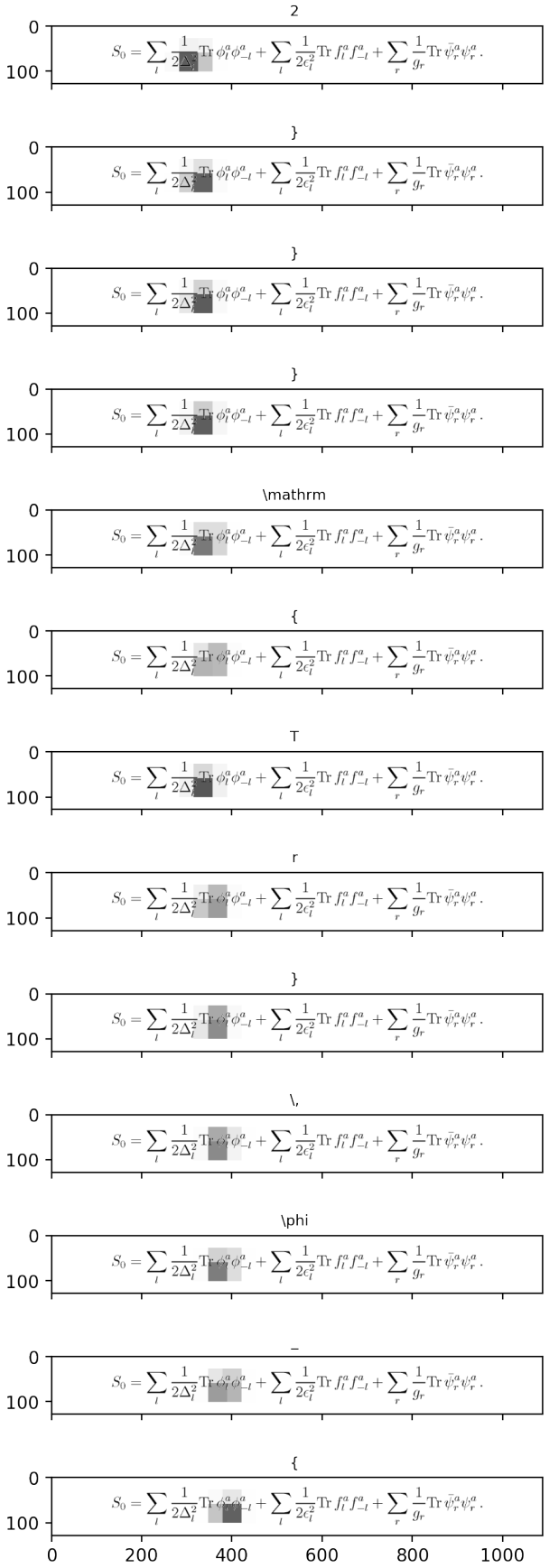

Figure 8. Attention scan at steps 25-37; continuing from Figure 5. The movement of focal-region is aligned with the output word (above the image).

taking an average of the feature vectors was causing a loss of information (which we mitigated by taking in all the L feature vectors without summing them). After making all the changes mentioned above, the Init Model yields a consistent albiet small performance improvement (Table. 10). But given that it consumes 7.5 million parameters, its usefulness is doubtful.

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

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

A.4. Training and Dataset

Default values of $\beta_1$andβ_2 of the ADAM optimizer - 0.9 and 0.99 yielded very jagged 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. $\beta_1$andβ_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 and raising the minimum-word-frequency threshold from 24 to 50 did yield better generalization and overall test scores (Table 7). Finally, normalizing the data (Section 2.5) yielded about 25% more accuracy than without.

Dataset	Model	Init Model?	λ_A	β_1	Training Epochs	Validation BLEU@Epoch	Training BLEU	Validation ED	$\overline{ASE_N}$
I2L-140K	I2L-STRIPS	Yes	0.0	0.5	104	0.8900@72*	0.9361	0.0677	5.3827
	I2L-STRIPS	No	0.0	0.5	75	0.8874@62	0.9300	0.0691	4.9899
	I2L-NOPOOL	Yes	0.0	0.5	104	0.8909@72*	0.9333	0.0684	4.5801
	I2L-NOPOOL	No	0.0	0.1	119	0.8820@92	0.9348	0.0738	4.7099
Im2latex-90k	I2L-STRIPS	Yes	0.0	0.5	110	0.8886@77*	0.9366	0.0688	5.1237
	I2L-STRIPS	No	0.0005	0.5	161	0.8810@118	0.9386	0.0750	4.8291

Table 11. Training parameters of a few runs. $\lambda_R = 0.00005$ and $\beta_2 = 0.9$ for all runs. Training BLEU scores were calculated over 100 mini-batches after decoding the sequences using CTC-decoding (Graves et al., 2006).

	$\boldsymbol{y}$	$\hat{\boldsymbol{y}}$
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	$\{\Phi^i(x),\Phi^j(y)\}=\epsilon^{ij}\,\delta^2(x-y).$	$\left\{\Phi^i(x),\Phi^j(y)\right\}=\epsilon^{ij}\,\delta^2(x-y).$
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	$\alpha^{\dagger a}_{\lambda}(p)=\int d^3x\; e^{-ip\cdot x}\left[e^{\lambda}\cdot(\omega A^a-iE^a)+\int_\Omega(f_1\Pi^a+f_2\phi^a)\right]$	$\alpha^{\dagger\alpha}_{\lambda}(p)=\int d^3x\; e^{-ip\cdot x}\left[e^{\lambda}\cdot(\omega A^a-iE^a)+\int_\Omega(f_1\Pi^a+f_2\phi^a)\right]$
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	$S[\Phi]=S[\phi]+S[\varphi]+S_{\rm int}[\phi,\varphi]$	$S[\Phi]=S[\phi]+S[\varphi]+S_{\rm int}[\phi,\varphi]$
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	$\chi(x_1,x_2)=\langle 0\|T\psi(x_1)\bar\psi(x_2)\|P\rangle.$	$\chi(x_1,x_2)=\langle 0\|T\psi(x_1)\bar\psi(x_2)\|P\rangle.$
12	$\epsilon L^{(2)}\theta=\tilde h_1^{(2)}v^2\phi^j(\epsilon\gamma^i\theta)(\theta\gamma^{ij}\theta)+h_1^{(2)}v^iv^j\phi^k(\epsilon\gamma^i\theta)(\theta\gamma^{jk}\theta),$	$\epsilon L^{(2)}\theta=\tilde h_1^{(2)}v^2\phi^j(\epsilon\gamma^i\theta)(\theta\gamma^{ij}\theta)+h_1^{(2)}v^iv^j\phi^k(\epsilon\gamma^i\theta)(\theta\gamma^{jk}\theta),$
13	$\sin^2 2\vartheta=\sin^2 2\vartheta_{\rm sun}=\frac{4\,\|U_{e1}\|^2\,\|U_{e2}\|^2}{(\|U_{e1}\|^2+\|U_{e2}\|^2)^2}\;.$	$\sin^2 2\vartheta=\sin^2 2\vartheta_{\rm sun}=\frac{4\|U_{e1}\|^2\|U_{e2}\|^2}{(\|U_{e1}\|^2+\|U_{e2}\|^2)^2}\;.$
14	$F_1=\frac{g^2}{192\pi^{5/2}}M_{Pl}\simeq 1.5{\times}10^{15}\;\text{GeV}.$	$F_1=\frac{g^2}{192\pi^{5/2}}M_{Pl}\simeq 1.5{\times}10^{15}\;\text{GeV}.$
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	$\bar h_v\,\Gamma\,h_v=\frac{1}{2}\,\mathrm{Tr}\left(\Gamma\,P_v\right)\bar h_v\,h_v-\frac{1}{2}\,\mathrm{Tr}\left(\gamma_\mu\gamma_5\,P_v\,\Gamma\,P_v\right)\bar h_v\,\gamma^\mu\gamma_5\,h_v\;,$	$\bar h_v\,\Gamma\,h_v=\frac{1}{2}\,\mathrm{Tr}(\Gamma\,P_v)\bar h_v\,h_v-\frac{1}{2}\,\mathrm{Tr}(\gamma_\mu\gamma_5\,P_v\,\Gamma\,P_v)\bar h_v\,\gamma^\mu\gamma_5\,h_v\;,$
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	$S\le S_{\rm H},\;\;T\ge T_{\rm H},\;\;\;E_c\le E_{\rm BH},\;\;\text{for}\;HR\ge 1$	$S\le S_{\rm H},\;\;T\ge T_{\rm H},\;\;\;E_c\le E_{\rm BH},\;\;\text{for}\;HR\ge 1$
21	$\Gamma\sim N^{-1/2}10^{23}\mathrm{s}^{-1}\exp\left[-\frac{8\sqrt{2}}{3{\cdot}137}\left(\frac{-E}{m_e}\right)^{3/2}\frac{B_0}{NB}A^{1/2}\left(\frac{m_p}{m_e}\right)^{1/2}\right],$	$\Gamma\sim N^{-1/2}10^{23}\mathrm{e}^{-1}\exp\left[-\frac{8\sqrt{2}}{3{\cdot}137}\left(\frac{-E}{m_e}\right)^{3/2}\frac{B_0}{NB}A^{1/2}\left(\frac{m_p}{m_e}\right)^{1/2}\right],$
22	$\frac{2\|J\|^2}{m^{0\,2}}\cong\frac{{m_b}^2}{{m_d}^2}\sim 2.5{\times}10^5\;.$	$\frac{2\int\|^2}{m^{0\,2}}\cong\frac{m_b^2}{m_d^2}\sim 2.5{\times}10^5\;.$
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	$\hat D^{-1}_{\mu\nu}=\hat D^{-1}_{\mu\alpha}\eta^{\alpha\beta}\left(\eta_{\beta\nu}+\sum_{n=1}^\infty A_n(D^{-1})^n_{\beta\nu}\right),$	$\hat D^{-1}_{\mu\nu}=\hat D^{-1}_{\mu\alpha}\eta^{\alpha\beta}\left(\eta_{\beta\nu}+\sum_{n=1}^\infty A_n(D^{-1})^n_{\beta\nu}\right),$
35	$H_G(x^2)=-\frac{1}{8\pi G\epsilon^2}\left([B(x^2)]^{-2}-1\right),$	$H_G(x^2)=-\frac{1}{8\pi G\epsilon^2}\left([B(x^2)]^{-2}-1\right),$
36	$T_{\mu\nu}=T^+_{\mu\nu}+\frac{a_-^2}{a_+^2}T^-_{\mu\nu}.$	$T_{\mu\nu}=T^+_{\mu\nu}+\frac{a_-^2}{a_+^2}T^-_{\mu\nu}.$
37	$G_{H^{d+1}}=\frac{l^{1-d}}{\Sigma_d}\int_{\frac{r}{l}}^{+\infty}\frac{dx}{\sinh^d x}\;.$	$G_{H^{d+1}}=\frac{l^{l-d}}{\Sigma_d}\int_q^{+\infty}\frac{dx}{\sinh^d x}\;.$
38	$\Gamma^{(n)}_{\{\mu\}}=\frac{\delta^{(n)}\Gamma(A')}{\delta A'_{\mu_1}(x_1)\cdots\delta A'_{\mu_j}(x_j)\cdots\delta A'_{\mu_n}(x_n)},$	$\Gamma^{(n)}_{\{\mu\}}=\frac{\partial^{n(n)}\Gamma(A')}{\delta A'_{\mu_1}(x_1)\cdots\delta A'^{o}_{\mu_\mu}(x_j)\cdots\delta A^o_{\mu_n}(x_n)},$
39	$\dot\pi_{\rm ab}=$	$\dot\pi_{\rm ab}=$
40	$\gamma\pi^0_a=0,\;\gamma\pi^i_a=f^b{}_{ac}\pi^i_b\eta^c_2,\;\gamma\pi_{0i}=0,\;\gamma\pi_{ij}=0,$	$\gamma\pi^0_a=0,\;\gamma\pi^i_a=f^b{}_{ac}\pi^i_b\eta^c_2,\;\gamma\pi_{0i}=0,\;\gamma\pi_{ij}=0,$
41	$\|Z_1\|^2=\|Z_2\|^2=\frac{1}{(4G)^2}e^{-\eta_0}\,[(Q_R{}^1)^2+(Q_R{}^2)^2]\;,$	$\|Z_1\|^2=\|Z_2\|^2=\frac{1}{(4G)^2}e^{-\eta_0}\,[(Q_R{}^1)^2+(Q_R{}^2)^2]\;,$

Table 12. A random sample of predictions of 12L-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]