Learning Fine-Grained Image Representations for Mathematical Expression Recognition

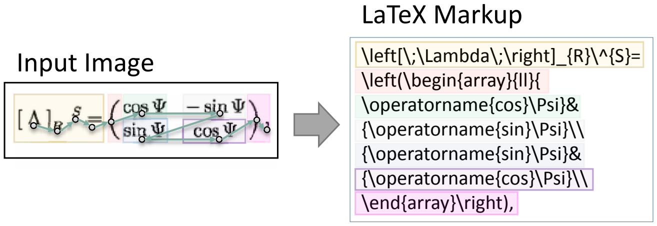
Sidney Bender* Monica Haurilet* Alina Roitberg Rainer Stiefelhagen
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany
sidney.bender@student.kit.edu, {haurilet, alina.roitberg, rainer.stiefelhagen}@kit.edu

***Abstract*—Optical character recognition is a key step towards automatically converting printed documents into electronic form. In this work, we consider the specialized task of mathematical expression recognition, which is characterized by a highly structured format and strict syntax rules, where the slightest mistake can lead to a very different meaning of the formula. To tackle this problem, we present a neural architecture based on a convolutional neural network focused specifically on fine-grained structures in the image. The obtained visual representations are used as an input to an encoder and an attention-based decoder module, trained jointly in an end-to-end manner. Given an input image, our model generates the underlying LaTeX markup that is able to perfectly describe the target mathematical formula. We conduct a thorough analysis of our model by examining the performance for different formula lengths and visualizing the attention maps of prediction examples. We demonstrate the effectiveness of our approach on the large-scale IM2LATEX-100K benchmark for mathematical expression recognition, where our model is able to outperform state-of-the-art methods, surpassing them by over 4% in image absolute accuracy.**

***Index Terms*—Offline Mathematical Expression Recognition, Deep Learning, Convolutional Neural Networks**

I. Introduction

Optical Character Recognition (OCR) is a key task in document analysis, aimed at text understanding from visual data. While conventional OCR approaches consider *natural language* recognition in images, mathematical expression recognition deals with automatically interpreting *mathematical formulas*. This task differs from plain text understanding in multiple important aspects. First, mathematical expressions have very strict syntax rules and even the smallest mistake can lead to extreme consequences in the markup (*e.g.* forgetting to close a bracket). Furthermore, causal connections between the symbols go far beyond left-to-right relations, ranging from a variety of horizontal (*e.g.* sub- and super-script) to vertical ones (*e.g.* fractions and matrices), as demonstrated in Figure 1. Such semantic relations are especially hard, as some of them cause the incorporated characters to be rescaled into a smaller size (*e.g.* sup- and superscript), while others enlarge them (*e.g.* matrix brackets). In this paper, we design a deep-learning based architecture specifically for mathematical expressions recognition, addressing these characteristic challenges of formulas by putting focus on *fine-grained* visual structures.



```
\left[\;\Lambda\;\right]_{R}\^{S}=
\left(\begin{array}{ll}{
\operatorname{cos}\Psi}&
{\operatorname{sin}\Psi}\\
{\operatorname{sin}\Psi}&
{\operatorname{cos}\Psi}\\
\end{array}\right),
```

Fig. 1: Example of an image and its underlying markup in the LaTeX format. In comparison to optical character recognition, where we solely have left to right ordering, in this example we can have difficult orderings both vertically and horizontally.

Fueled by the progress of deep learning, computer vision problems (*e.g.* image classification [1]) and high-level natural language understanding tasks (*e.g.* question answering [2]) experienced tremendous success in recent years. In this work, we introduce a neural architecture for recognizing mathematical expressions from images, without the necessity of any additional annotations except for the underlying LaTeX markup (*e.g.* we do not need segmentations labels of the individual symbols). We tackle the variable-sized symbols by representing the image using fine-grained features that are thereafter embedded using a module for relation encoding. We produce the underlying markup in the widely-used and easily understandable markup language LaTeX, which is popular in fields like Mathematics, Computer Sciences and Physics [3] and is also often used in schools and universities by users with visual impairment.

We evaluate our approach on the large-scale IM2LATEX-100K dataset [4] that consists of 100K annotated images and show that our model is able to improve state-of-the-art. Furthermore, we analyze the performance of our model for expression lengths (ranging from only 10 symbols to 500 markup characters), showing better recognition rates for shorter formulas. To obtain a better interpretation of our model, we examine portions of the image which contributed most to the prediction by visualizing the weights of the model generated by our in-built attention module.

In summary, the contributions of our work are as follows:

- We present an architecture for mathematical expression recognition based on deep neural networks, that uses a

*indicates equal contribution

978-1-7281-5054-3/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDARW.2019.00015

novel *fine-grained* CNN with a recurrent decoder and an attention module over the image regions.
- We analyze our proposed model in detail by evaluating its performance for different expression lengths.
- We present qualitative results by visualizing example predictions with their corresponding attention maps.
- Finally, we demonstrate the effectiveness of our model on the large-scale IM2LATEX-100K dataset, surpassing the current state-of-the-art approaches by a large margin.

II. Related Work

Optical Character Recognition. Optical Character Recognition (OCR) deals with the translation of an image into an underlying pre-defined text representation in natural language format. Various models for OCR rely on pre-defined feature representations of the underlying data [5]–[7] and classifiers *e.g.* conditional random fields [8], [9], support vector machines or k-nearest neighbor. Deep learning approaches were able to strongly improve previous methods for OCR [10]. Encouraged by this success, we also use a deep neural architecture for tackling the related task of off-line Mathematical Expression Recognition (MER). Nevertheless, our model has to tackle other difficulties than OCR methods, as mathematical expressions (MEs) have a wider range of possible relations *i.e.* both horizontal as well as vertical relationship types.

Machine Translation. Machine translation deals with mapping a given sequence into a different variable-sized series (*e.g.* translating text from German to French). In order to solve the unknown alignment problem between the n words of the input sentence and the m words of the corresponding translation, Cho *et al.* [11] introduced the encoder-decoder model. The encoder-decoder model uses a Long Short-Term Memory (LSTM) [12] to encode the input sentence into a single vector. A second LSTM is then used to decode this vector into the variable-sized target series.

The single global representation of the entire sequence produces a bottleneck, as the entire *variable-sized* information is compressed into a single *fixed-sized* representation. Furthermore, since LSTMs also tend to loose information of the input with increasing time step, [13] make use of attention mechanisms for the decoder module. During the decoding process, these attention modules allow the LSTM to focus on parts of the image and, thus, handling both the bottleneck of a fixed-sized representation as well as the problem when using a high number of time steps. Since we also have to deal with a high number of steps (some MEs are over 150 tokens long), we use an attention mechanism in the decoder module. However, in comparison to machine translation, our input is not a variable-sized vector but a variable-sized 2D array representation of a ME.

Image Captioning. Image captioning is a related task to machine translation that instead of translating an input sequence, it generates sentences based on an input image. As image captioning is similar to the machine translation task, we see similar tools that were used for machine translation also applied for image captioning. Xu *et al.* [14] successfully extend this attention-based encoder-decoder-model to image captioning by using a CNN block as the encoder and an attention-based LSTM similar to [13] as the decoder. In this work, we use a similar deep learning method that consists of a feature extractor, an encoder and an attention-based decoder. A key difference of MER to image captioning is the length of the predicted sequence, since generally in image captioning only short sequences of about 10 words are produced, while in MER we have to handle MEs of over 150 tokens.

Off-line Mathematical Expression Recognition. Recognizing mathematical expressions from images is a popular task in document analysis tackled already by various previous works [4], [15]–[17]. Methods based on pre-defined features [18] are based on a three step approach: 1) character segmentation, 2) symbol classification, and 3) structural analysis. The first step consists of segmenting each individual symbol, often using clustering or connected component analysis. Each found segment is classified into one of the pre-defined class symbols (*e.g.* α, $\sum$), using *e.g.* nearest neighbor, which maps every connected component to a probability distribution over the known classes. Finally, possible parse trees are produced via a two dimensional stochastic Cocke-Younger-Kasami (CYK, [19]) algorithm, where the tree with the highest probability is chosen as the final mathematical expression.

In a similar way, INFTY Reader [17] performs the following steps: 1) image parsing (detection and classification of symbols), 2) graph generation (connecting related symbols) and 3) structural analysis (producing the final tree by extracting the minimum spanning tree of the previously generated graph). The final representation consists of the tree in XML format that can be converted into other types of markup formats.

In [4], a novel deep learning architecture for printed mathematical expression is introduced that represents the image using a coarse convolutional neural network, which is further encoded using an LSTM. Finally, to produce the ME, a second recurrent neural network with an attention mechanism is used. In comparison, our model makes use of a *fine-grained* feature extractor and a *fine-grained* attention map, that as we will show, offers a better representation of the input image as it is able to focus on even very small symbols in the formula.

III. Neural Network Architecture

In this section, we introduce a neural network architecture for mathematical expression recognition (overview in Figure 2). First, the model embeds the input image containing a mathematical formula using the *fine-grained* feature extractor FGFE (Figure 3). The produced features are then passed to the encoder, which maps them to 1) local features and to 2) a global representation of the image. Based on the global representation and the local features the decoder constructs the predicted formula via an LSTM with an attention mechanism step-by-step. The formula thereby is represented as LaTeX tokens of atomic size *e.g.* ‘a’, ‘\begin{array}’ or ‘\frac’.

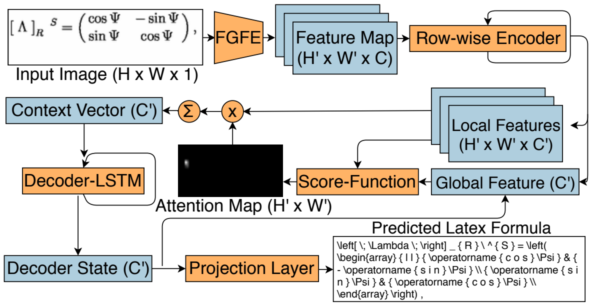

$[\Lambda]_R{}^S = \begin{pmatrix} \cos\Psi & -\sin\Psi \\ \sin\Psi & \cos\Psi \end{pmatrix},$
Input Image (H x W x 1)
FGFE
Feature Map (H' x W' x C)
Row-wise Encoder
Context Vector (C')
Σ
x
Local Features (H' x W' x C')
Decoder-LSTM
Score-Function
Global Feature (C')
Attention Map (H' x W')
Predicted Latex Formula
Decoder State (C')
Projection Layer
\left[\; \Lambda \; \right] _ { R } \ ^ { S } = \left(\begin{array} { l l } { \operatorname { c o s } \Psi } & { - \operatorname { s i n } \Psi } \\ { \operatorname { s i n } \Psi } & { \operatorname { c o s } \Psi } \\ \end{array} \right) ,

Fig. 2: Our neural architecture for generating mathematical expressions based on an input image. The core modules of our model are: 1) a feature extractor based on a CNN for obtaining the feature maps; 2) a relational encoder that embeds the feature cells into a new representation; 3) the decoder generates the LaTeX markup using the embeddings produced by the relational encoder.

We choose this token-wise format over a character based representation, as this avoids irregular results and reduces the formula length.

A. The Fine-Grained Feature Extractor (FGFE)

Our fine-grained feature extractor consists of a Convolutional Neural Network (CNN) that is able to compute useful high-level features from the preprocessed input gray-scale image of size $H \times W$. Depending on the number of layers and especially the size of the stride of each convolutional or pooling layer, the receptive field and thereby the granularity can be adjusted. While in case of image captioning on natural images, large receptive fields are conventional used [1], we argue that for mathematical expression recognition as we have very small symbols, a fine-grained representation benefits the neural network.

As we see in Figure 3, FGFE consists of three CNN blocks which consist of 2-4 convolution layers, a max-pooling layer and, finally, batch-normalization is applied. In order not to reduce the size of the image too fast and thus keep the fine-grained structure of the ME, 2×1 and 1×2 max-pooling layers are applied with a stride of 2×1 and 1×2 in the second and the third block, respectively. For an input image of dimensions $H \times W$, our FGFE produces an output of the shape $H/4 \times W/4 \times 128$ *i.e.* only slightly reducing the image size.

B. Encoder

To translate a variable-sized three dimensional feature map into a formula, we make use of an encoder-decoder-module. The encode operation maps the visual feature map to a global feature and a refined local representation. An LSTM is used row-wise on the feature map and then, by concatenating the outputs of the rows we obtain the final local representation. For the global feature, we average all final states of each row of the encoder LSTM.

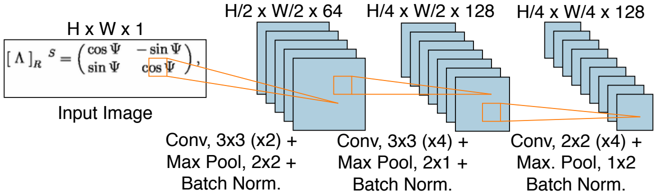

H x W x 1
$[\Lambda]_R{}^S = \begin{pmatrix} \cos\Psi & -\sin\Psi \\ \sin\Psi & \cos\Psi \end{pmatrix},$
Input Image
H/2 x W/2 x 64
H/4 x W/2 x 128
H/4 x W/4 x 128
Conv, 3x3 (x2) + Max Pool, 2x2 + Batch Norm.
Conv, 3x3 (x4) + Max Pool, 2x1 + Batch Norm.
Conv, 2x2 (x4) + Max. Pool, 1x2 Batch Norm.

Fig. 3: The Fine-Grained Feature Extractor (FGFE) consists of three blocks: the first consists of two convolution layers, while the latter ones contain four convolution layers each.

C. Attentional Decoder

Finally, we use the local and global visual representations to generate the underlying markup by applying an LSTM with attention mechanism. In every step T, we get a score value for every local feature using an additive scoring function, which compares the local feature with the current state of the decoder LSTM (see Figure 2). We model a probability distribution over the local features by normalizing these scores using softmax and, then, we use them as weights for the local features. Next, we average these weighted local embeddings and obtain a single fixed-dimensional context vector that is fed together with the current decoder state into the decoder LSTM cell. The projection layer then maps the new decoder state to a probability distribution over the token classes. During prediction, we select the token class with the highest activation in each decoding step T.

D. Learning Setting

We minimize the cross entropy loss using stochastic gradient decent on the IM2LATEX-100K dataset for 15 epochs. As [4], we start with an initial learning rate of 0.1 and decay it by 10 every time the validation loss does not improve. For a stable training procedure, we feed the embedding of the correct token from the last decoding step into the decoder LSTM. During test time, we use beam search [20], with a width of 5 for decoding in order to reduce search errors.

IV. Evaluation

In this section, we compare our model to related work based on both token- and image-based evaluation metrics. Then, we discuss the performance of our models for different formula lengths and show that even though the accuracy drops for very large expressions, we are still able to often produce correct MEs. We analyze the performance for different formulas based on their structure types *e.g.* matrices and fractions, and the impact of rare token classes on performance. Finally, we provide some example predictions with the corresponding attention maps.

A. Evaluation Metrics

To evaluate our models, we use the same token- and image-based metrics as related work [4]. While *token-based metrics* compare the predicted token sequence $pred$ with the ground truth markup gt, *image-based metrics* use images $predImg$ and $gtImg$, which were generated by the LaTeX compiler.

Approach	Attention	BLEU	EDA_T	ABS_T	EDA_I	ABS_I
Classical Methods and Baselines						
Prior	–	0.0	20.0	0.0	85.0	0.0
INFTY [17]	–	66.7	–	–	–	26.7
Deep Learning Architectures						
CTC [22]	–	30.4	–	–	–	9.2
Caption [14]	softmax	75.0	–	–	–	55.7
Im2Tex [4]	hierarch.	86.2	–	–	–	79.6
	hard	87.1	–	–	–	77.1
	sparsemax	87.0	–	–	–	78.1
	softmax	87.7	92.1	41.2	88.6	79.9
Ours	softmax	**90.3**	**92.8**	**46.8**	**93.1**	**84.3**

TABLE I: Test Accuracies on the IM2LATEX-100K dataset based on five metrics. We group the methods into two categories: deep-learning and non-deep-learning methods.

Absolute Token Accuracy (ABS_T) is an evaluation metric that calculates the accuracy of exact matches between the ground truth gt and prediction $pred$ in LaTeX format.

Token Edit Distance Accuracy (EDA_T) softens the ABS_T by rewarding predictions that are similar to the ground truth:

$$EDA_T(pred, gt) = 1 - \frac{distance(pred, gt)}{max(|pred|, |gt|)}, \qquad (1)$$

where $distance$ denotes the Leventhstein [21] distance between $pred$ and gt, and $|x|$ denotes the number of tokens of the sequence x. The Leventhstein distance is defined as the number of edit operations (remove, swap, add) that are necessary to map $pred$ to gt.

Absolute Image Accuracy (ABS_I) is similar to the token-based absolute accuracy where we have a correct prediction if $predimg$ and $gtimg$ match exactly. As in related work [4], since white spaces have no semantic meaning in mathematical expressions, for the comparison we remove columns consisting of only of whitespace in $predimg$ and $gtimg$.

Image Edit Distance Accuracy (EDA_I) is calculated by first converting the images $predimg$ and $gtimg$ to arrays of booleans based on the threshold 0.5. The final EDA_I is defined as the average row-wise Leventhstein distance as depicted in Eq. 1.

BLEU is a popular metric used to evaluate distances between sentence pairs in natural language (*e.g.* in image captioning). BLEU is defined as the geometric mean of BLEU-1 to BLEU-4 times a brevity penalty. BLEU-n thereby calculates the n-gram overlap between both sentences (*i.e.* precision), while the brevity penalty penalizes short sentences (*i.e.* recall).

B. Overall Results on IM2LATEX-100K

In this section, we provide the final results of our model on the test set of the large-scale IM2LATEX-100K dataset (see Table I). Since the LaTeX compiler is not an injective function (*i.e.* different LaTeX markups can generate identical images), a text-only metric can have difficulties to provide a good comparison between mathematical expressions. Thus, we provide results for both token- as well as image-based metrics.

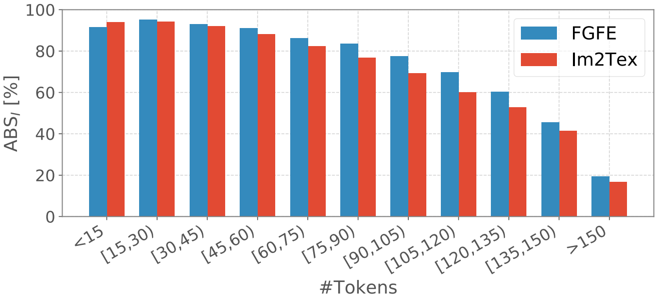


Fig. 4: Overview of the impact of formula length on performance. While short formulas almost achieve perfect accuracy (over 95%), long formulas still have room for improvement.

Table I shows the results of our prior baseline, which always produces the most frequent tokens in the training data. More precisely, in case of the image-based metrics, the prior model produces an image containing the most frequent pixel *i.e.* only white pixels. For the token-based metrics the prior classifier always predicts a formula of the average formula length from the training data *i.e.* consisting of 52 tokens. The first half contains only left curly brackets tokens, as these are at the first positions the most frequent ones, while in the second half the most frequent token is the right curly bracket. Not surprisingly, as we have a very large number of different classes the baseline shows 0% for $BLEU$, ABS_I and ABS_I, while in case of the easier EDA_T and EDA_I we achieve a significantly better performance of about 20% and 85%, respectively.

The INFTY reader applied to mathematical expression recognition easily beats our prior baseline by a large margin (*i.e.* from 0% to 26.7% ABS_I), while CTC improves the performance of the baseline by 9%. In comparison, the Caption method which does not use an encoder module, shows a further improvement compared to INFTY.

The Im2Tex deep learning method that uses both an encoder and an attention-based decoder, is able to show very promising results improving the Caption network by over 20% ABS_I. As we see in Table I, different attention mechanisms are evaluated on the Im2Tex architecture: 1) a hierarchical attention that combines different layer outputs of the encoder, 2) hard and 3) sparse max which both use discretized values and, thus, have to be trained using reinforcement learning. Nonetheless, the popular 4) softmax normalization shows the best performance in comparison the other proposed attention modules, having an overall improvement of over 50% over the widely used INFTY model in Image Absolute Accuracy.

Finally, our model achieves state-of-the-art results of 84.3% in Image Absolute Accuracy outperforming Im2Tex by over 4% and the Caption model by almost 30%. We want to note that the gap between our model and related work is much greater in absolute accuracies than in the edit distance metrics. This is not surprising, as the edit distance metrics are by far easier, as we also see the small gap in edit distance of our prior baseline to the other approaches. Nonetheless, we were able to improve the results in the edit distances as well as the popular text-based BLEU metric.

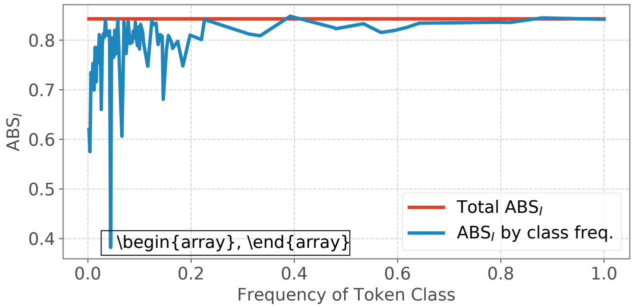


Fig. 5: Impact of rare token classes on the absolute image accuracy of our model.

C. Impact of the Formula Length on Performance

In Figure 4, we analyze the impact of formula length on the image absolute accuracy of our neural architecture. As expected, we experience a drop in performance for longer MEs, *e.g.* for formulas under 100 tokens we achieve an accuracy of 80%, while for MEs over 150 tokens we obtain around 20%. We also note that very short formulas (*i.e.* shorter than 15 tokens) have a worse performance than medium-sized one (*i.e.* between 15 and 30 characters). The reason for these results is probably due to the imbalance in the training data, since short formulas are by far less frequent (under 3% of MEs) than the longer ones (over 13% of MEs have a length between 15 and 30 tokens).

D. Impact of Rare Token Classes

In Figure 5, we show the relation between the average ABS_I of images containing a token and the frequency of the particular token classes. Thus, the Y-axis shows the absolute image accuracy of all images containing the particular token class at least once. The X-axis corresponds to its frequency *i.e.* percentage of images that contain at least one appearance of the particular token class in the ground truth data.

We notice that there is a strong correlation between the frequency of the token classes and the prediction performance of our model: the fewer training samples we have at our disposal the lower is our accuracy. Moreover, formulas which contain at least one of the 61 worst recognized token classes, are never predicted correctly and have a frequency of less than 1% in the training set. The reason that ABS_I does not drop to 0% when reaching rare tokens with a frequency of nearly 0 is that many symbols have exact synonyms (*i.e.* predicting the exact synonym instead of a rare token does not impact ABS_I).

E. Importance of a Fine-grained Visual Representation

In Figure 6, we show an example attention map that was produced by our model (top) and by the architecture in related work [4] (bottom). Since the formulas contain sub- and superscripts, many characters are smaller than the default token size. Nonetheless, the decoder using the attention maps of FGFE is able to concentrate its weights to the small delta character, leading to a correct prediction of the '\Delta' token. In comparison, the decoder applied on the coarser features of related work, was not able to focus on a single character, but on multiple characters simultaneously. Thus, this merge of features of multiple characters lead the model to produce erroneously an '\alpha' token.

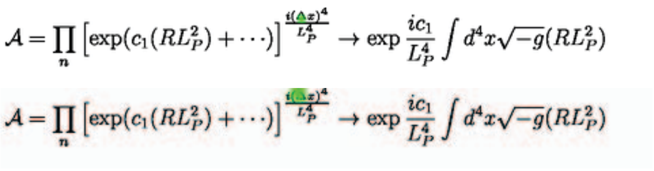


Fig. 6: Comparison of the attention maps of FGFE when generating the token '\Delta' (top) and of Im2Tex when incorrectly producing the token '\alpha' (bottom).

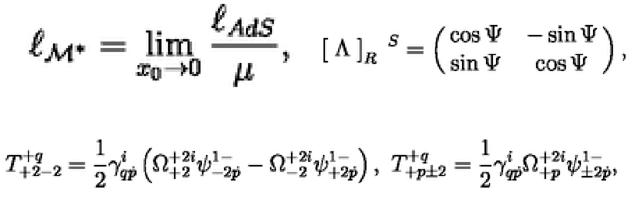


Fig. 7: Examples of correctly predicted MEs containing arrays, subscripts, superscripts and fractions.

F. Analysis of Highly-Structured Mathematical Expressions

Highly-structured MEs still constitute a problem for the model, as we show in Figure 5 where a strong drop in performance occurs due to MEs of the matrix type *i.e.* arrays. Even though array-tokens have around 5K example images, the network has difficulties in learning these complex structures. Nonetheless, the model is often able to correctly predict such complex structures (see examples in Figure 7).

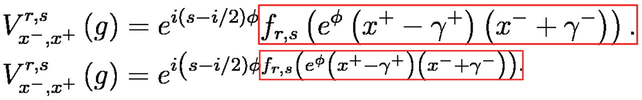


Fig. 8: Example of misrecognition of the baseline: ground truth (top) and prediction (bottom).

G. Discussion about Typical Errors

Additionally to difficulties on highly-structured MEs, we encounter other two types of errors: 1) simple errors caused by brackets and 2) a subsequence of the formulas are generated multiple times.

Simple Errors caused by Brackets. In Figure 8, we show an example where the model included a pair of brackets which led to a wrong prediction. In this example, the model closed the bracket too late and, thus, the entire second half of the formulas was placed in the exponential of the Euler number. Even though not all our errors are this simple, we notice many such examples where only few characters are included or removed from our prediction. Thus, leading to a strong drop in both image edit distance and image absolute accuracy.

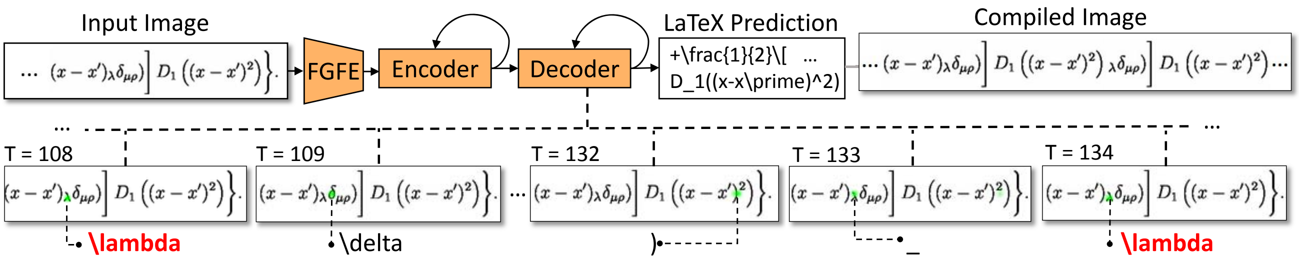


Fig. 9: Recursive behavior of the model for long MEs. This example shows correctly predicted tokens up to time step $T = 132$. For $T = 133$, the model loops back to the previously generated lambda causing a recursion in the prediction.

Recursive Behavior in Long Formulas. In some long MEs, the network produces infinite many times a subsequence of the formula *i.e.* the model stops only when the maximum number of time steps is reached. In Figure 9, we show an example of a ME on which the model generates an infinite loop of the following subsequence: ‘${}_{\lambda}\sigma_{\mu\rho})]D_1((x - x\prime)^2)$’. After predicting the final round bracket, the model jumps back to generating the lambda from previous steps. When visualizing this behavior we see that even the attention module focuses on the previously predicted token (*i.e.* the lambda in our example). The probable culprit of these loops is the decoder, which is not able to keep track on the already generated tokens. Nonetheless, these errors occur only on very long formulas, where the decoder LSTM has to memorize a large number of previous steps.

V. Conclusion

In this work, we developed a deep learning architecture for mathematical expression recognition. We use a novel fine-grained feature extractor that is trained in an end-to-end manner with a relational encoder and a decoder with an attention mechanism using softmax normalization. Since the symbols have a large range of different sizes, a small receptive field in the feature extractor benefits our model, especially as we use an encoder on the produced feature maps that is able to link regions of larger characters. Thus, we show that our model is able to learn nontrivial spatial relations like fractions and arrays (see Figure 7). Furthermore, the model is able to predict some remarkably large formulas, which are longer than formulas contained in the training set. Our model shows strong results on all metrics outperforming current state-of-the-art by over 4% in absolute image accuracy.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems (NIPS)*, 2012.

[2] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, “End-to-end memory networks,” in *Advances in neural information processing systems (NIPS)*, 2015.

[3] F. Brischoux and P. Legagneux, “Dont format manuscripts,” *The Scientist*, 2009.

[4] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, “Image-to-markup generation with coarse-to-fine attention,” in *International Conference on Machine Learning (ICML)*, 2017.

[5] T. E. De Campos, B. R. Babu, M. Varma *et al.*, “Character recognition in natural images.” *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009.

[6] B. Epshtein, E. Ofek, and Y. Wexler, “Detecting text in natural scenes with stroke width transform,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[7] L. Neumann and J. Matas, “A method for text localization and recognition in real-world images,” in *Asian Conference on Computer Vision (ACCV)*, 2010.

[8] A. Mishra, K. Alahari, and C. Jawahar, “Top-down and bottom-up cues for scene text recognition,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[9] J. J. Weinman, E. G. Learned-Miller, and A. R. Hanson, “A discriminative semi-markov model for robust scene text recognition.” in *International Conference on Pattern Recognition (ICPR)*, 2008.

[10] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, “End-to-end text recognition with convolutional neural networks,” in *International Conference on Pattern Recognition (ICPR)*, 2012.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, 1997.

[13] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[14] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in *International Conference on Machine Learning (ICML)*, 2015.

[15] R. H. Anderson, “Syntax-directed recognition of hand-printed two-dimensional mathematics,” in *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, 1967.

[16] K.-F. Chan and D.-Y. Yeung, “Mathematical expression recognition: a survey,” *International Journal on Document Analysis and Recognition (IJDAR)*, 2000.

[17] M. Suzuki, T. Kanahori, N. Ohtake, and K. Yamaguchi, “An integrated ocr software for mathematical documents and its output with accessibility,” *Computers Helping People with Special Needs*, 2004.

[18] F. Alvaro, J.-M. Benedi *et al.*, “Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011.

[19] J.-C. Chappelier, M. Rajman *et al.*, “A generalized cyk algorithm for parsing stochastic cfg,” *Workshop on Tabulation in Parsing and Deduction*.

[20] S. Abdou and M. S. Scordilis, “Beam search pruning in speech recognition using a posterior probability-based confidence measure,” *Speech Communication*, 2004.

[21] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, 1966.

[22] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *International Conference on Machine learning*, 2006.