A Transformer Architecture for Online Gesture Recognition of Mathematical Expressions

Mirco Ramo[1,2(✉)] and Guénolé C. M. Silvestre[2]

[1] Dip. Ingegneria dell'Informazione, University of Pisa, Pisa, Italy
`mirco.ramo@ucdconnect.ie`
[2] School of Computer Science, University College Dublin, Dublin, Ireland
`guenole.silvestre@ucd.ie`

Abstract. The Transformer architecture is shown to provide a powerful framework as an end-to-end model for building expression trees from online handwritten gestures corresponding to glyph strokes. In particular, the attention mechanism was successfully used to encode, learn and enforce the underlying syntax of expressions creating latent representations that are correctly decoded to the exact mathematical expression tree, providing robustness to ablated inputs and unseen glyphs. For the first time, the encoder is fed with spatio-temporal data tokens potentially forming an infinitely large vocabulary, which finds applications beyond that of online gesture recognition. A new supervised dataset of online handwriting gestures is provided for training models on generic handwriting recognition tasks and a new metric is proposed for the evaluation of the syntactic correctness of the output expression trees. A small Transformer model suitable for edge inference was successfully trained to an average normalised Levenshtein accuracy of 94%, resulting in valid postfix RPN tree representation for 94% of predictions.

Keywords: Online gesture recognition · Transformer · Multilevel segmentation · Expression tree · Transfer learning · RPN

1 Introduction

Modern edge communicating devices are built around touch-sensitive display panels equipped with handwriting recognition systems. These systems are of great assistance eschewing the need for structured UIs such as virtual keyboards that are often slow and error-prone while also distant to the natural handwriting experience with pens.

In this context, online recognition of glyphs (as opposed to offline that takes a graphical image representation as input) refers to the problem of mapping spatio-temporal samplings of user gestures corresponding to handwritten text into a symbolic representation. Each 3-dimensional sample individuates a touch. A coherent and consecutive sequence of touches defines a stroke that can be combined to form glyphs. Glyphs correspond to characters or symbols encoded

Table 1. Terminology

Term	Definition
Touch/point	$(x, y, [t])$ tuple of finger location on touch panel sampled at time t
Stroke	Sequence of points where finger consecutively touches the panel
Glyph	List of one or more strokes individuating an element in the vocabulary
Numeral/operand	Ordered list of glyphs corresponding to digits $\{0, \ldots, 9\}$ and comprising at most one decimal notation mark $\{.\}$ – note: a numeral always evaluates in a finite rational numeric value
Symbol	One of the 4 operators $\{+, -, \times, \div\} \cup \{=, (,)\}$

in a language vocabulary. In this work, we will consider the online input of mathematical arithmetic expressions as a formally correct sequence of gestures of numerals, operators and symbols. Note that some numerals and operators may require more than one stroke to be represented as depicted in Fig. 1. Table 1 formalizes the terminology adopted in this work.

Gesture recognition applications must solve several problems at once, namely: i) feature extraction in a multi-dimensional spatio-temporal space, ii) segmentation of stroke sequences into glyph items, iii) glyph segmentation with the aim of numeral recognition, and iv) the encoding of expression rules and patterns to form a correct symbolic output. An example of an online gesture sequence is shown in Fig. 1.

With mathematical expressions, users often wish to go beyond the mere recognition of glyphs and hope for additional tasks to be performed such as automatic evaluation, step-by-step simplification or listing of equivalent forms. The Expression Tree (ExpTree) formalism [1] was introduced to represent mathematical expressions as binary trees and consequently resolve all equivalent forms to some unique representation, scheduling its evaluation by transforming an input symbol list into a computation graph. In particular, a post-ordered traverse of tree generates the Reverse Polish Notation (RPN) using postfix notation, a unique representation that postpones operators, crushing the need for brackets.

Main Contributions

(i) We propose a new dataset for handwritten expressions (cf. Sect. 3) obtained from several hundred users and suitable for a wide range of supervised and unsupervised machine learning applications.

(ii) We study the ability of an attention mechanism to learn and represent implicit structures of spatio-temporal gesture data, even when the underlying syntax is not enforced (in the loss computation or model architecture).

(iii) We prove the power of Transformers not only as language models but also as a solution to several sequence mapping tasks, demonstrating transfer
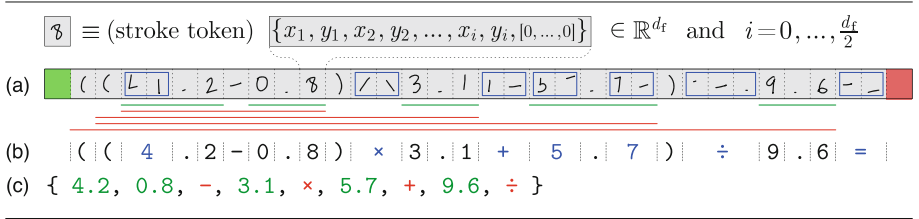
$\boxed{\text{%}} \equiv (\text{stroke token}) \quad \boxed{\{x_1, y_1, x_2, y_2, ..., x_i, y_i, [0, ..., 0]\}} \quad \in \mathbb{R}^{d_f} \quad \text{and} \quad i = 0, ..., \frac{d_f}{2}$

(a)

(b) ((4 . 2 − 0 . 8) × 3 . 1 + 5 . 7) ÷ 9 . 6 =

(c) { 4.2, 0.8, −, 3.1, ×, 5.7, +, 9.6, ÷ }

Fig. 1. Online gesture example of a stroke sequence (a) for the mathematical expression (b) and its corresponding RPN expression tree (c). Each cell in (a) depicts the linear interpolation of spatio-temporal points that forms an input token. Green and red cells denote the ⟨bos⟩ and ⟨eos⟩ respectively. Glyph segmentation, numeral segmentation and RPN ExpTree parsing are colour coded with blue, green and red respectively. (Color figure online)

> learning behaviours of the encoder on unseen glyphs from online gesture input[1].

(iv) We propose a small footprint[2] topology for end-to-end online mathematical expression recognition and ExpTree generation, with fast optimisation, very high accuracy and suitable for edge inference.

(v) We test the model robustness on ablated input, showing its ability to generate compliant RPN expressions even in case of missing strokes.

(vi) We show the multi-level segmentation capability of the attention mechanism, highlighting the correlation between syntactically correct predictions and explainability in cross-attention visualisation.

2   Related Work

The field of Handwriting Text Recognition (HTR) consists on a set of techniques and algorithms that aim at generating text directly from handwritten inputs. Most HTR systems [2] work on offline data due to dataset availability [3]. With the current popularity of the attention mechanism [4,5], the field remains in constant development. However, as noted in [6], the temporal dimension provides some valuable additional information that may simplify stroke segmentation and avoid recourse to complicated regression strategies such as text-line segmentation [7]. As a result, online methods may expect superior performance over offline counterparts as reported in a 2014 extensive survey of online HTR methods [6]. Further progress has since been observed, with much effort and resources employed on improving existing techniques [8,9].

---

[1] Encoder with frozen parameters (pre-trained on digit-agnostic datasets) subsequently used on a new task, taking token input from spatio-temporal sequences in a potentially infinitely large vocabulary.

[2] Despite its small footprint, model can perform the tasks of glyph segmentation, numeral segmentation, character recognition and tree building at remarkable performance levels, learning efficiently the input/output mapping.

In this context, Handwriting Digit Recognition (HDR) remains a popular HTR sub-problem still actively researched using both offline [10] and online methods [11]. In particular Handwritten Mathematical Expression Recognition (HMER) consists in the generation of mathematical expressions using formal syntaxes such as LATEX. State-of-the-art HMER models have reached impressive levels of accuracy, particularly when exploiting attention [12] and combining potentialities of online and offline data [13]. However, although predictions are mostly correct, these models fail to learn the intrinsic structure of the mathematical expression. In contrast, learning a tree representation provides a more natural form [14] and can be achieved with an RNN encoder and a HMER tree decoder to explicitly represent the tree formalism.

We propose to push this challenge further, leaving the task of learning implicitly the RPN syntax to the model, and doing this by relying on the attention mechanism embedded in a Transformer framework [15]. This provides a powerful sequence mapping architecture entirely based on the attention mechanism [4], eschewing recourse to recurrent or convolution layers, hence allowing for significant parallelisation and unattenuated gradient flow. This topology currently stands as the state-of-the-art on almost all NLP tasks [16–18], but also on a wider and more generic group of sequence transduction problems [19–23]. The Transformer popularity saw many experiments revisiting its design with several optimized architectures being proposed [24–27]. However very few are capable of clearly outperforming the original topology. As a result this work will follow the seminal Transformer proposal of [15].

3   Dataset

An important contribution of this work is that of an online gesture dataset of mathematical expressions suitable for investigating several tasks such as Handwriting Character Recognition (HCR), HDR or HMER, but also touch, stroke or glyph segmentation, automatic result computation, unsupervised generation or eventually, ExpTree building. Our handwritten database is presented as a coherent collection of tables composing a SQL Schema with spatio-temporal data for arabic numerals [11] and mathematical symbols, collected from volunteers writing on touch panels. This stage saw the contribution of 455 subjects for a total of 21 752 labelled glyphs composed by 27 477 strokes, thus over 700 thousand touches. The dataset can be used at different levels of granularity, namely *touch*, *stroke* and *glyph*.

Subjects have been split into training, validation and test sets (60/20/20 proportions) such that models were tested on unseen handwriting styles to ensure accurate estimation of the generalisation power. In addition, strokes were also randomly augmented and composed to form expressions.

An expression (E) is defined as a bounded sequence of numerals (N) and operators ($\odot \in \{+, -, \times, \div\}$). The generation of expressions is carried out according to the following grammar:

1. an expression can be a numeral: $E \rightarrow N$

2. an expression can be a binary operation: $E \to E \odot E$
3. an expression can be a binary operation between brackets: $E \to (E \odot E)$

As a supplementary rule, every expression must end with the '=' symbol. For each expression, we provide 3 ground truth labels (namely ASCII text, RPN tree and numerical evaluation), for a total of 240 000 samples split as specified above. In this work, we report results at the stroke level, leaving to the model the burden of glyph segmentation.

4   Transformer Architecture and Experimental Details

Our model leverages the original Transformer architecture [15]. However crucial modifications are introduced to work with spatio-temporal data. Given some input sequence, $X \in \mathbb{R}^{d_f \times n}$, of n stroke tokens defined as interleaved spatio-temporal data with zero-padding of fixed-length d_f (maximum of 64 (x, y) touch samples per stroke, appropriately $\langle\texttt{bos}\rangle$ prefixed, $\langle\texttt{eos}\rangle$ suffixed and $\langle\texttt{pad}\rangle$ padded), a mask M_x is computed to ensure encoder's attention is only paid on valid online data tokens.

As the input is composed of spatio-temporal information corresponding to touches, each encoder token embeds a stroke as d_f scalars (cf. Fig. 1) resulting in the identification within a potentially unbounded input vocabulary and therefore eschewing any form of embedding.

Positional encoding provides a strategy to embed the positional information of input tokens in the encoder, a necessary operation since the attention mechanism has no built-in concept of sequentiality. Frequency modulation is proposed in [15]. However, since we observed no performance gain with such a strategy, we use a learnable 1D embedding based on the incremental index of the token. Stroke positions are encoded in $P_x \in \mathbb{R}^{d_f \times n}$.

The encoder is trained to learn some latent sequence representation $Z = \mathrm{Enc}(X + \alpha P_x, M_x) \in \mathbb{R}^{d_a \times d_h \times n}$ where α is a scaling factor blending the input data and positional information, d_a the number of attention heads and d_h the hidden state dimension of the attention heads. The encoder consists of a stack of l_e identical multi-head vanilla self-attention layers and a positional feed-forward network of dimension d_p. Each layer is followed by a residual connection before layer-normalisation.

In this work, we explored the transfer learning capabilities of the encoder that was never trained from scratch but relied on an optimised snapshot, pre-trained in conjunction with a language modelling decoder using a large corpus of English sentences [28] that contained almost no digits and arithmetic operators (classified as $\langle\texttt{unk}\rangle$ tokens). This transfer learning strategy resulted in considerable speed-up during training and model optimisation. We use a frozen encoder with Θ_e parameters as a feature extractor on this new domain.

The decoder generates a causal sequence of tokens in an auto-regressive manner given some vocabulary and relative token encoding. It is initialised with the

Table 2. Expression recognition, model hyper-parameters and dataset configuration. Performance is reported in term of Cross-Entropy Loss (XEL) and normalised Levenshtein distance Accuracy (LA). Model v_4 trained on larger expressions using 4 Heads (H) in a 4 Layer (L) decoder performs best.

Model	Data size	Bracket	Enc* (l_e, d_a)	Dec (l_d, d_a)	XEL	LA (%)
v_1	10k	No	5L, 4H	2L, 4H	0.76	71.3
v_2	240k	No	5L, 4H	4L, 4H	0.43	84.8
v_3	240k	No	5L, 4H	4L, 2H	0.43	84.8
v_4	240k	Yes	5L, 4H	4L, 4H	**0.40**	**84.9**

*Encoder: using transfer learning with frozen (untrained) parameters

⟨bos⟩ token and iteratively outputs a new token using greedy sampling of the decoder's softmax output until the ⟨eos⟩ token is predicted or the maximum sequence length, m, is reached. The decoder also consists of l_d identical layers, each composed by: i) a masked self-attention layer that prevents the decoder from peeking at the subsequent tokens, ii) a cross-attention layer that attends over the encoder output Z to generate predictions, and iii) a feed-forward layer as in the encoder but of dimension $3d_p$.

At each step, the decoder's input is an auto-regressive sequence of tokens mapped into an embedding layer with positional encoding, and used to predict the next token of the output sequence. All Θ_d parameters of the decoder were trained from some randomly initialised state.

Experimental details: models were all configured with $d_f = d_a \times d_h = d_p = 128$. For v_{1-5}, $n = 2m = 24$. For v_{10-11}, $n = 2m = 48$. The encoder has $\Theta_e = 523\,520$ parameters and decoder has $\Theta_d = 934\,136$ parameters. Models were trained on Nvidia TitanX GPUs[3], for a maximum of 200 epochs, using cross-entropy loss and Adam optimiser with a decay schedule (initial learning rate of 8×10^{-4} and halving every 30 epochs).

5   Experimental Results

A series of experiments were carried out to investigate two different problems, namely: (1) expression recognition in glyph sequences and (2) ExpTree recognition in RPN forms. The first task involves the recognition of a sequence of glyphs composing an arithmetic expression from stroke input as time series. The second task requires further understanding of symbolic syntax and semantics through the construction of an ExpTree using postfix notation.

Models are evaluated using a number of performance metrics on the test sets and results are reported in terms of: (a) Cross-Entropy Loss (XEL), (b) normalised Levenshtein distance [29] Accuracy (LA), (c) Character Error Rate (CER), and where applicable (d) RPN Accuracy Range (cf. Sect. 5.2). The LA and CER are both accuracy metrics based on edit distance.

Table 3. ExpTree recognition for various model hyper-parameters. Performance is reported in term of Cross-Entropy Loss (XEL), normalised Levenshtein distance Accuracy (LA), Character Error Rate (CER), and RPN Accuracy Range (RAR). Models trained on 240k expression datasets. Fine-tuned model v_{11} with $\langle$eon$\rangle$ for numeral segmentation provides best performance.

Name	Enc (l_e, d_a)	Dec (l_d, d_a)	XEL	LA(%)	CER	RAR (%)
v_5	5L, 4H*	4L, 4H‡	0.46	83.7	0.19	$[91.8, 91.8]$
v_{10}	5L, 4H*	4L, 4H‡	0.34	87.4	0.14	$[92.4, 93.2]$
v_{11}	5L, 4H†	4L, 4H†	**0.24**	**93.7**	**0.07**	$[\mathbf{93.3}, \mathbf{94.0}]$

*Frozen parameters
†Fined-Tuned parameters using transfer learning
‡Trained from scratch from some randomly initialised state

5.1   Expression Recognition

In this set of experiments, models v_{1-3} are trained to output glyph sequence of simple arithmetic expressions in the absence of brackets while model v_4 adds groups of terms with brackets. Table 2 summarises training datasets, model hyper-parameter configuration and performance evaluation in this experiment.

We observe that there are no clear benefits in increasing the number of decoder heads in the absence of brackets (models v_{2-3}). However, despite an increase of vocabulary size and, in principle, also some decoding complexity, the addition of brackets resulted in better performance as seen in model v_4. This model is capable of learning some non-trivial valuable syntax rules such as number of '(' should match that of ')', or an operator can never precede a ')'.

5.2   Expression Tree Recognition

The ExpTree recognition task requires of an additional step to glyph recognition with the construction of an RPN form. In this set of experiments, model performance is also evaluated in terms of CER and RPN Accuracy Range (RAR) defined as the range $[1 - V_\ell^{\mathsf{max}}, 1 - V_\ell^{\mathsf{min}}]$, where V_ℓ stands for violation loss. If v_i denotes the count of violations in the i-th expression, $V_\ell^{\mathsf{min}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{v_i > 0}$ and $V_\ell^{\mathsf{max}} = \frac{1}{N} \sum_{i=1}^{N} v_i$, where N is the test set cardinality. Referring to the standard infix to postfix conversion algorithm in [1], a violation occurs every time the stack is in an inconsistent state while conversion is performed.

This does not required the initialisation of stack operations to be determined. Instead one can linearly scan the output using a counter, incrementing its value for a push, decrementing it for a pop. Counter value should be 1 at the end and never become negative. Adding the number of times a negative value is observed to the absolute value of the final counter minus 1 defines the number of violations.

Table 3 summarises experimental results on ExpTree predictions. Models $v_{5, 10-11}$ were trained on the same dataset size as v_{2-4} (240k expressions), with
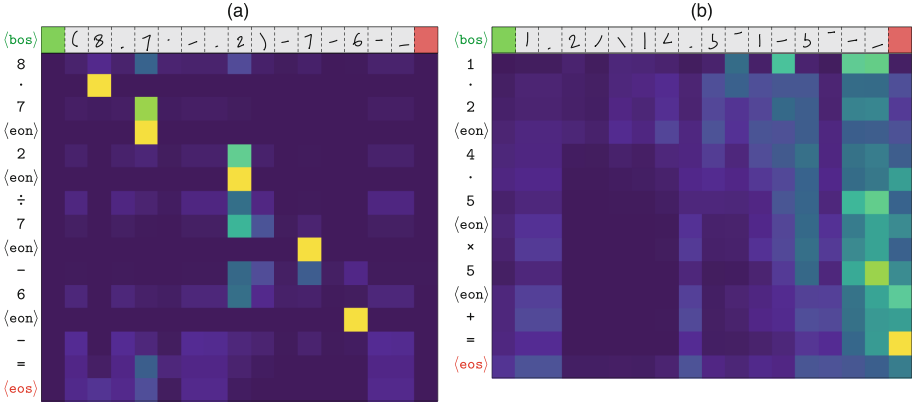
Fig. 2. Cross-attention plots. In (a), output tokens '.' (decimal mark) and ⟨eon⟩ (end-of-numeral) can be seen tracking the previous digit; in (b), output token '=' is attending token ⟨eos⟩.

the possible inclusion of brackets. The v_5 model training dataset further constrained numerals to contain at most one decimal digit. This restriction was lifted in training sets associated with models v_{10-11}. As a result, an end-of-numeral token, ⟨eon⟩, was added to the decoder's output vocabulary for learning an additional numeral segmentation task of RPN forms.

With the same hyper-parameter configuration of Table 2, an expected degradation in performance is observed for model v_5 on this more complicated task. The addition of the ⟨eon⟩ token in model v_{10} showed some significant improvement in accuracy, outperforming our best results for simple expression glyph recognition. Despite the use of a larger vocabulary size for the decoder's output, the addition of a specific token to model explicitly the language semantic of numerals is observed to yield higher accuracy once again. The new token forces the network to learn a pattern resulting in better numeral segmentation and improved performance.

In Sect. 4 we proposed to test the transfer learning capabilities of the encoder, using frozen parameters on a new domain. Excellent results have been observed, demonstrating the encoder's ability to correctly segment and combine strokes generating latent representations that are generic enough to be valuable for any downstream tasks even when used with completely different output vocabulary.

However, further improvement can be reached with fine-tuning of all parameters as observed with model v_{11} that leveraged frozen encoder weights of model v_{10}, introducing the concepts of digits or operators for the first time. Final model achieves 94% on the Normalised Levensthein Accuracy, with a Character Error Rate lower than 7%, generating on average 94% of strings compliant to the RPN, while mean number of violations per output expression is only 0.067.
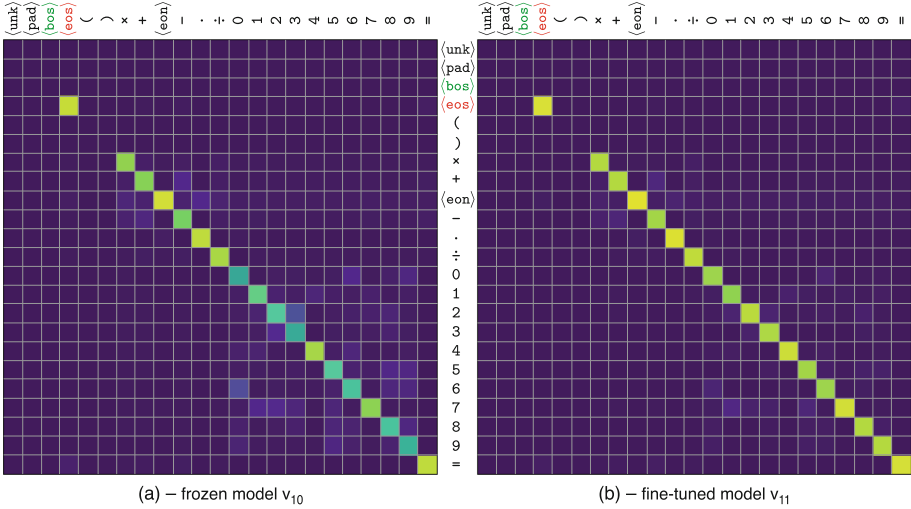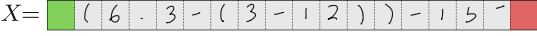
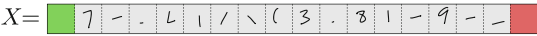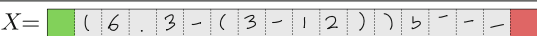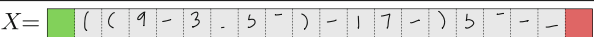(a) – frozen model v_{10} (b) – fine-tuned model v_{11}

Fig. 3. Softmax distribution mean of the decoder's output predictions showing the probability mass for all token pairs. Frozen model v_{10} in plot (a) reveals decoding errors caused by confusion between digits 2 and 3, and also between operators. Model v_{11} in plot (b) shows that fined-tuning on all glyphs reduces confusion dramatically.

6 Attention Visualisation and Output Distributions

Visualisation of attention mechanisms provides some interesting insights in the learning process. Figure 2a depicts the cross-attention weights that the decoder puts over the encoder's output. It shows that head 1 of layer 1 is responsible for numeral segmentation. For every ⟨eon⟩ or decimal mark tokens, this head has learned to attend over the stroke of the previous digit. In Fig. 2b, head 4 of layer 3 attends over the ⟨eos⟩ token while predicting the '=' token demonstrating that the model has successfully learned the syntax rule 'every expression must end with '=' symbol'.

Figure 3 shows the confusion matrix over the decoder's vocabulary for the average probability distribution of the output softmax. This provides some insight into model mispredictions leading to errors. In Fig. 3a, model v_{10} leveraged a frozen encoder pre-trained on a completely different output vocabulary with no digits and operators. The model confuses '2' with '3' and, to a lesser extent, operator '-' with '+' since the latter is often written with an horizontal stroke. Figure 3b shows that fine-tuning the encoder in model v_{11} results in better performance and improved diagonality, which also justifies the greedy decoding strategy used in our decoder.

Table 4. Model robustness: ablation experiments with input strokes elided from input and corresponding to the equal sign in rows 1–3, a closing bracket in row 4 and an operator in rows 5–6. Metric is the Levenshtein Distance (LD).

Model	Input (X), Ground truth (Y) & Model inference ($\hat{Y}$)	LD
v_4	$X=$ ▢ L ı / ＼ 6 ▢ $Y = \{\langle\text{bos}\rangle\, 4 \,\times\, 6 \,\langle\text{eos}\rangle\}$ $\hat{Y} = \{\langle\text{bos}\rangle\, 4 \,\times\, 6 \,{=}\, \langle\text{eos}\rangle\}$	1
v_{10}	$X=$ ▢ (6 . 3 - (3 - ı 2)) - ı 5 ⌐ ▢ $Y = \{\langle\text{bos}\rangle\, 6 . 3 \,\langle\text{eon}\rangle\, 3 \,\langle\text{eon}\rangle\, 2 \,\langle\text{eon}\rangle\, {+}\,{-}\, 5 \,\langle\text{eon}\rangle\, {+}\, \langle\text{eos}\rangle\}$ $\hat{Y} = \{\langle\text{bos}\rangle\, 6 . 2 \,\langle\text{eon}\rangle\, 3 \,\langle\text{eon}\rangle\, 2 \,\langle\text{eon}\rangle\, {+}\,{-}\, 5 \,\langle\text{eon}\rangle\, {+}\,{=}\, \langle\text{eos}\rangle\}$	2
v_{11}	$X=$ ▢ 9 ı - ((7 - · - . 3) - 2) ▢ $Y = \{\langle\text{bos}\rangle\, 9 \,\langle\text{eon}\rangle\, 7 \,\langle\text{eon}\rangle\, 3 \,\langle\text{eon}\rangle\, {\div}\, 2 \,\langle\text{eon}\rangle\, {-}\,{+}\, \langle\text{eos}\rangle\}$ $\hat{Y} = \{\langle\text{bos}\rangle\, 9 \,\langle\text{eon}\rangle\, 7 \,\langle\text{eon}\rangle\, 3 \,\langle\text{eon}\rangle\, {\div}\, 2 \,\langle\text{eon}\rangle\, {-}\,{+}\,{=}\, \langle\text{eos}\rangle\}$	1
v_4	$X=$ ▢ 7 - . L ı / ＼ (3 . 8 ı - 9 - _ ▢ $Y = \{\langle\text{bos}\rangle\, 7 . 4 \,\times\, (3 . 8 \,+\, 9 \,{=}\, \langle\text{eos}\rangle\}$ $\hat{Y} = \{\langle\text{bos}\rangle\, 7 . 4 \,\times\, (3 . 8 \,+\, 9) \,{=}\, \langle\text{eos}\rangle\}$	1
v_{10}	$X=$ ▢ (6 . 3 - (3 - ı 2)) 5 ⌐ - _ ▢ $Y = \{\langle\text{bos}\rangle\, 6 . 3 \,\langle\text{eon}\rangle\, 3 \,\langle\text{eon}\rangle\, 2 \,\langle\text{eon}\rangle\, {+}\,{-}\, 5 \,\langle\text{eon}\rangle\, {=}\, \langle\text{eos}\rangle\}$ $\hat{Y} = \{\langle\text{bos}\rangle\, 6 . 2 \,\langle\text{eon}\rangle\, 3 \,\langle\text{eon}\rangle\, 2 \,\langle\text{eon}\rangle\, {+}\,{-}\, 5 \,\langle\text{eon}\rangle\, {+}\,{=}\, \langle\text{eos}\rangle\}$	2
v_{11}	$X=$ ▢ ((9 - 3 . 5 ⌐) - ı 7 -) 5 ⌐ - _ ▢ $Y = \{\langle\text{bos}\rangle\, 9 \,\langle\text{eon}\rangle\, 3 . 5 \,\langle\text{eon}\rangle\, {-}\, 7 \,\langle\text{eon}\rangle\, {+}\, 5 \,\langle\text{eon}\rangle\, {=}\, \langle\text{eos}\rangle\}$ $\hat{Y} = \{\langle\text{bos}\rangle\, 9 \,\langle\text{eon}\rangle\, 3 . 5 \,\langle\text{eon}\rangle\, {-}\, 7 \,\langle\text{eon}\rangle\, {+}\, 1 \,\langle\text{eon}\rangle\, {-}\,{=}\, \langle\text{eos}\rangle\}$	2

7 Model Robustness

Model robustness is investigated by means of ablation studies; strokes are removed from the input sequence to observe the model's ability to enforce domain rules even when it is fed with incorrect expressions.

Equal sign ablation: in our dataset, every expression to be considered syntactically correct must end with '='. The learning of this rule is assessed by observing the inference results of models $v_{4,\,10-11}$ when strokes representing the equal sign are omitted in the encoder's input. All three models are able to make the correct inference, inserting the missing '=' in decoder's output as shown in rows 1–3 of Table 4.

Closing bracket ablation: in any correct plain expressions, the number of '(' should match that of ')'. This syntactic rule is investigated in model v_4

that was trained to recognise glyphs of an expression (not possible with models v_{10-11} as RPN forms eschew the use of brackets). When the stroke of a closing bracket was removed from the encoder's input, the model acknowledges the input error and inserts the missing bracket in the output as shown in row 4 of Table 4. Of course, the exact position is not always guessed correctly, but the symbol is predicted so that to ensure syntax correctness of the output.

Operator ablation is investigated on models v_{10-11}, where an operator's strokes is removed from the input as shown in rows 5–6 of Table 4. To ensure ExpTree correctness when using postfix notation, an output expression must be terminated by an operator and its total number of operators always be a unit lower than the cardinality of operands. Both models appear to have learned this rule and are able to infer the presence of additional operator at the end (actual operator can only be guessed).

8   Conclusion

This work proposed a Transformer network for mathematical expression tree building from online input gesture data corresponding to handwritten strokes of digits and mathematical symbols. The encoder's input was modified to receive spatio-temporal data as real-valued tokens. It can directly operate at stroke level without the need for mapping on a fixed input vocabulary. Model can predict ExpTrees by handling internally the multi-level segmentation of inputs (at glyph and numeral levels) and also understanding and learning how to represent and enforce syntactic and semantic rules of data. In addition, index positional encoding was shown to be as effective as cosine modulation yet standing as a simpler and more natural encoding for the position information. The Transformer's ability to generate complex representations and learn non-trivial input/output mapping between sequences is well known [16,19]. However the challenge was further pushed in this work with no ad hoc solutions to represent syntax or semantic rules and the absence of an engineered loss computation and model architecture. In addition, the encoder was trained on a completely different domain [28] and used as a frozen feature extractor in most experiments. Such transfer learning capabilities suggest that the encoder can create general latent representations suitable for problems of different nature, reducing the overall number of model parameters. The objective of this work is not so much to push out some state-of-the-art model but rather to state some important considerations that may be the starting points for future works in language modelling. Neural Machine Translation may be extended in this way to online data at different granularity levels, with no need for separate input segmentation or complex positional embeddings. Finally, pre-trained encoders could be effectively leveraged with transfer learning on different domains without fine-tuning or explicit domain adaptation, accelerating training for new problem classes where computational power/time or dataset size is limited.

References

1. Gries, D.: Compiler Construction for Digital Computers. Wiley, New York (1971)
2. Plamondon, R., Srihari, S.: Online and off-line handwriting recognition: a comprehensive survey. IEEE Trans. Pattern Anal. Mach. Intell. **22**(1), 63–84 (2000)
3. Sinwar, D., Dhaka, V.S., Pradhan, N., et al.: Offline script recognition from handwritten and printed multilingual documents: a survey. Int. J. Doc. Anal. Recogn. (IJDAR) **24**(1), 97–121 (2021)
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations (ICLR) (2015)
5. Poulos, J., Valle, R.: Character-based handwritten text transcription with attention networks. Neural Comput. Appl. **33**(16), 10563–10573 (2021). https://doi.org/10.1007/s00521-021-05813-1
6. Kim, J.H., Sin, B.-K.: Online handwriting recognition. In: Doermann, D., Tombre, K. (eds.) Handbook of Document Image Processing and Recognition, pp. 887–915. Springer, London (2014). https://doi.org/10.1007/978-0-85729-859-1_29
7. Barakat, B., Droby, A., Kassis, M., El-Sana, J.: Text line segmentation for challenging handwritten document images using fully convolutional network. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 374–379 (2018)
8. Keysers, D., Deselaers, T., Rowley, H.A., et al.: Multi-language online handwriting recognition. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1180–1194 (2017)
9. Graves, A.: Generating sequences with recurrent neural networks. arXiv (2013). https://doi.org/10.48550/arXiv.1308.0850
10. Shrivastava, A., Jaggi, I., Gupta, S., et al.: Handwritten digit recognition using machine learning: a review. In: 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC), pp. 322–326 (2019)
11. Corr, P.J., Silvestre, G.C., Bleakley, C.J.: Open source dataset and deep learning models for online digit gesture recognition on touchscreens. In: 2017 Irish Machine Vision and Image Processing Conference (IMVIP) (2017). https://doi.org/10.48550/arXiv.1709.06871
12. Li, Z., Jin, L., Lai, S., et al.: Improving attention-based handwritten mathematical expression recognition with scale augmentation and drop attention. In: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 175–180 (2020)
13. Wang, J., Du, J., Zhang, J., et al.: Multi-modal attention network for handwritten mathematical expression recognition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 1181–1186 (2019)
14. Zhang, J., Du, J., Yang, Y., et al.: SRD: a tree structure based decoder for online handwritten mathematical expression recognition. IEEE Trans. Multimed. **23**, 2471–2480 (2021)
15. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30, pp. 6000–6010 (2017)
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 4171–4186. Association for Computational Linguistics (2019)
17. Brown, T., Mann, B., Ryder, N., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020)

18. Wolf, T., Debut, L., Sanh, V., et al.: Transformers: state-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45 (2020). https://doi.org/10.18653/v1/2020.emnlp-demos.6
19. Parmar, N., Vaswani, A., et al.: Image transformer. In: International Conference on Machine Learning, pp. 4055–4064. PMLR (2018)
20. Huang, C.Z.A., Vaswani, A., et al.: Music transformer: generating music with long-term structure. In: International Conference on Learning Representations (ICLR) (2019)
21. Zhao, H., Jiang, L., Jia, J., et al.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16259–16268 (2021)
22. Kozlov, A., Andronov, V., Gritsenko, Y.: Lightweight network architecture for real-time action recognition. In: Proceedings of the 35th Annual ACM Symposium on Applied Computing, pp. 2074–2080 (2020)
23. D'Eusanio, A., Simoni, A., Pini, S., et al.: A transformer-based network for dynamic hand gesture recognition. In: 2020 International Conference on 3D Vision (3DV), pp. 623–632 (2020)
24. Wang, S., Li, B.Z., Khabsa, M., et al.: Linformer: self-attention with linear complexity. arXiv (2020). https://doi.org/10.48550/arXiv.2006.04768
25. Kitaev, N., Kaiser, L., Levskaya, A.: Reformer: the efficient transformer. In: International Conference on Learning Representations (2020)
26. Choromanski, K., Likhosherstov, V., Dohan, D., et al.: Rethinking attention with performers. In: International Conference on Learning Representations (2021)
27. Rao, R.M., Liu, J., Verkuil, R., et al.: MSA transformer. In: International Conference on Machine Learning, pp. 8844–8856. PMLR (2021)
28. Akinremi, O., Balado, F., Silvestre, G.C.: A machine translation model for online glyph recognition. UCD Internal Research Report (2021, to be published)
29. Yujian, L., Bo, L.: A normalized levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1091–1095 (2007)