A review of paper

Deep Learning for Symbolic Mathematics

Guillaume Lample, Francois Charton
Facebook AI Research

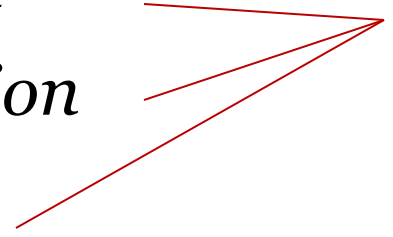
June 12, 2020

Published on ICLR 2020

BROWN

Background

- Symbolic Mathematics
  - *Symbolic differentiation*
  - *Symbolic integration*
  - *Symbolic simplification*
  - *Symbolic ODEs*



undecidable

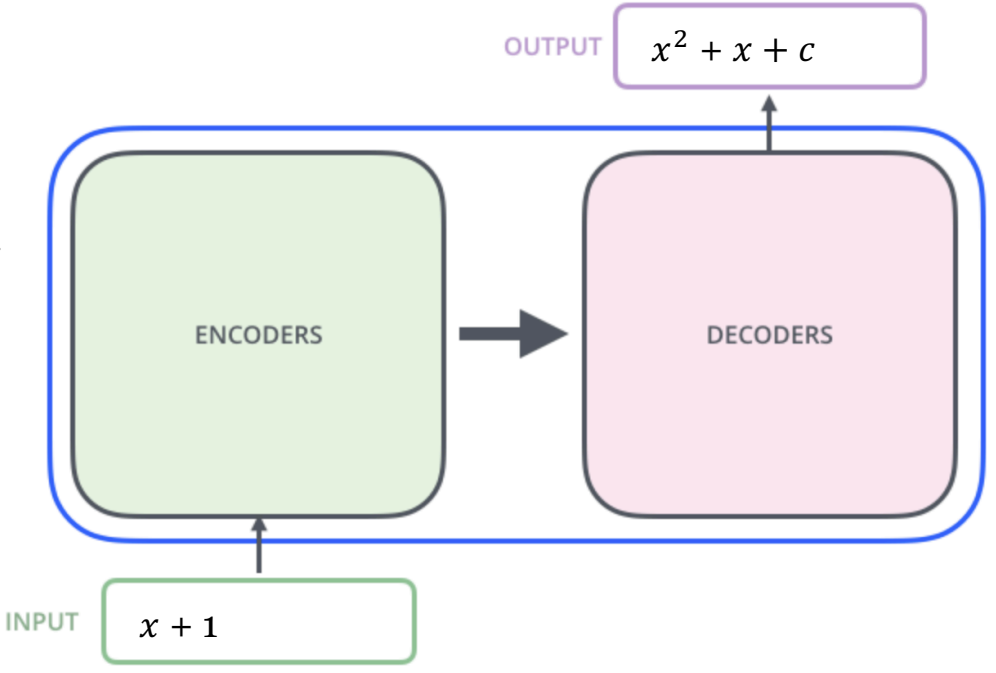
"Given an elementary expression, finding an elementary symbolic integral is, in general, a search in an enormous and strange state space for something that most of the time does not even exist. Even if you happen to know that it exists, it remains a very hard problem."

--Ernest Davis

BROWN

Background

- Seq2seq Modeling
  - *LSTM*
  - *Transformer*
  - *BERT, GPT…*



BROWN

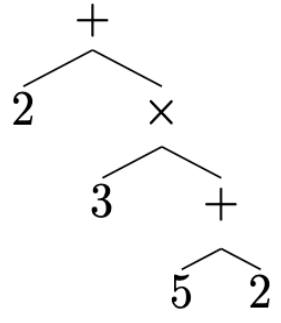
Methodology

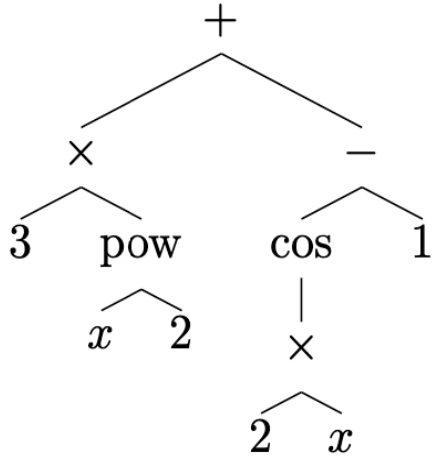
- A large corpus of examples (80 million) was created synthetically by generating random, complex pairs of symbolic expressions and their derivatives.
- A seq2seq transformer model is trained on the corpus.
- At testing time, A function g to integrate is fed into the model. An answer f produced by the model was checked using the following procedure: the symbolic differentiator was applied to f, and then the symbolic simplifier tested whether $f' = g$.

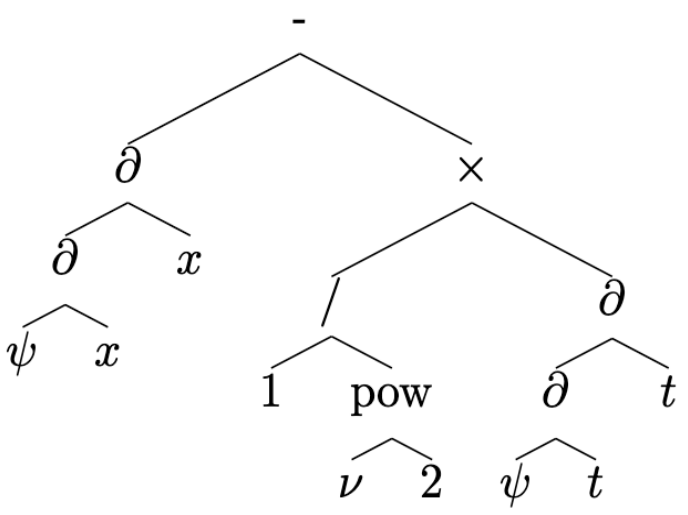
BROWN

Data Generation

- Expressions as trees(one to one)







$2 + 3\times(5 + 2)$

$3x^2 + \cos(2x) - 1$

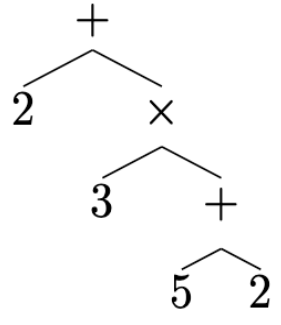
$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{v^2}\frac{\partial^2 \psi}{\partial t^2}$$

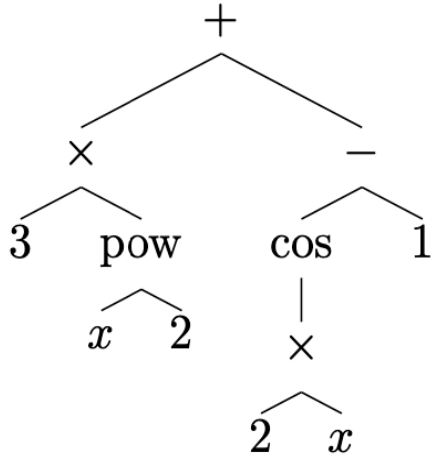
BROWN

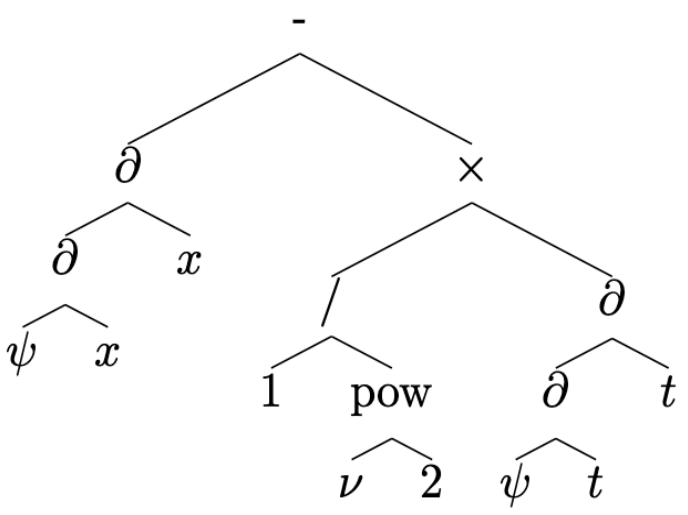
Data Generation

- Trees as sequences(one to one)

  Use prefix notation: write each node before its children, left to right







[+ 2 ∗ 3 + 5 2]

...

...

Data Generation

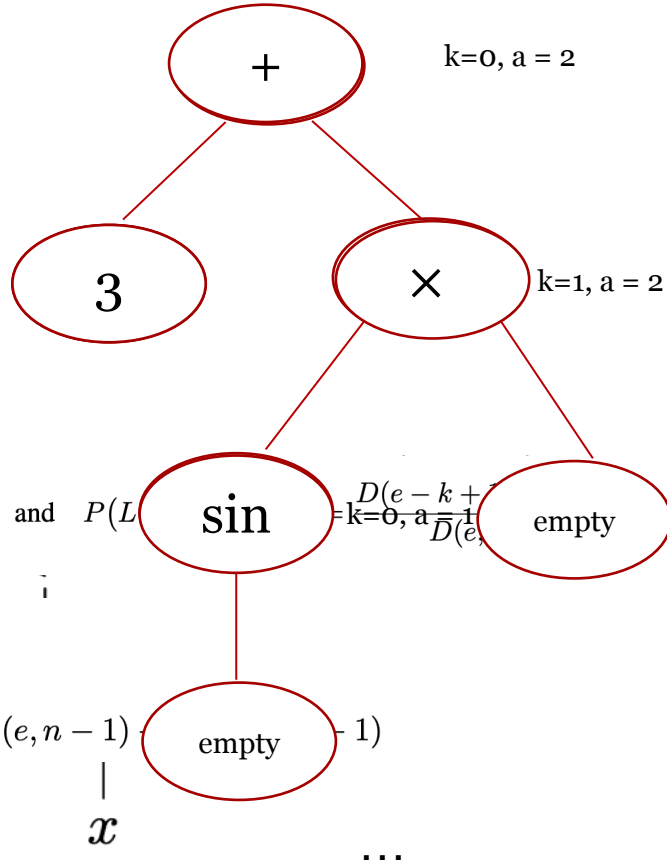
- Generate trees(unary-binary):

  Sample uniformly from trees with n internal nodes, n fixed.

Start with an empty node, set $e = 1$;
while $n > 0$ **do**
  Sample a position k and arity a from $L(e, n)$ (if $a = 1$ the next internal node is unary);
  Sample the k next empty nodes as leaves;
  if $a = 1$ **then**
    Sample a unary operator;
    Create one empty child;
    Set $e = e - k$;
  end
  else
    Sample a binary operator;
    Create two empty children;
    Set $e = e - k + 1$;
  end
  Set $n = n - 1$;
end

$$P(L(e,n) = (k,1)) = \frac{D(e-k, n-1)}{D(e,n)} \quad \text{and} \quad P(L$$

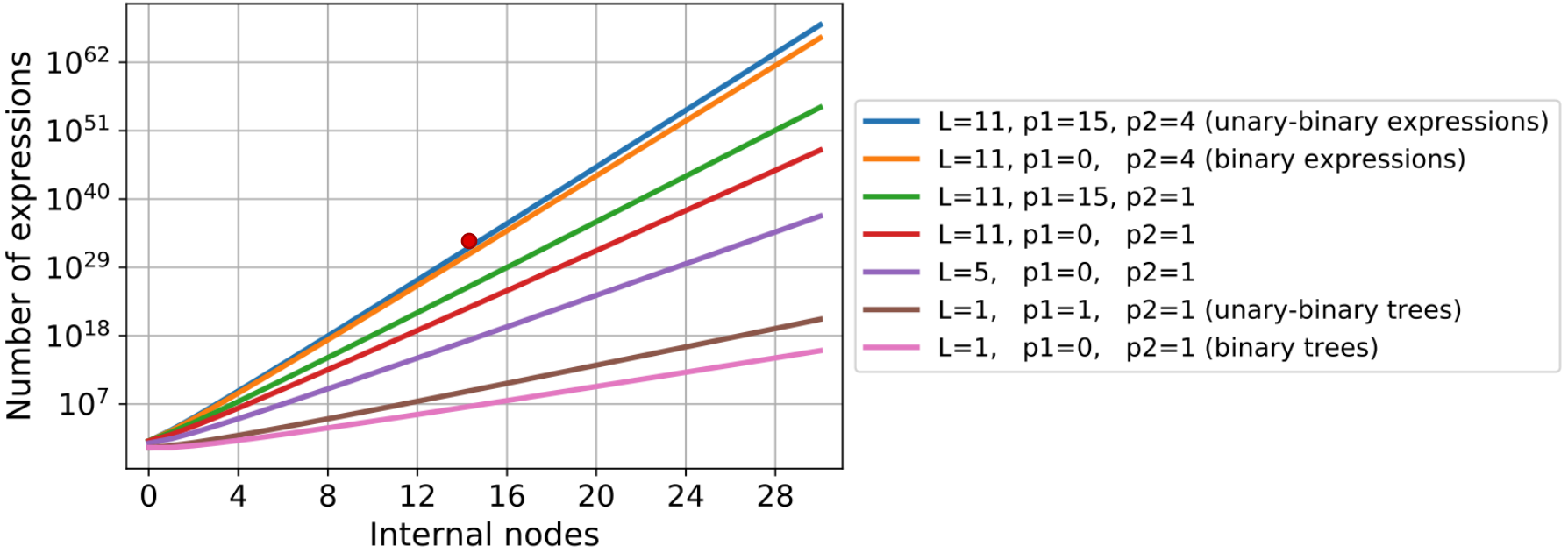
$$D(0,n) = 0$$
$$D(e,0) = 1$$
$$D(e,n) = D(e-1,n) + D(e,n-1)$$



x x ...

BROWN

Data Generation



BROWN

Data Generation

Task 1 : Integration

- Forward generation(FWD): Generate random sequences as input, then integrate with computer algebra system as output
- Backward generation(BWD): Generate random sequences as output, then differentiate as input
- Backward generation with integration by parts(IBP): Randomly sample F and G, get their derivative f and g respectively.

$$\int Fg = FG - \int fG$$

If $(Fg, \int Fg)$ is already in the training dataset, then we know $\int fG$, thus can add $(fG, \int fG)$ to the dataset. Vice Versa.
If none of them belongs to the dataset, then resample F and G.

BROWN

Data Generation

Task 2 : First-order ODEs (ODE 1)

output

Generate a random function	$f(x) = x \log(c \,/\, x)$
Solve in c	$c = xe^{\frac{f(x)}{x}} = F(x, f(x))$
Differentiate in x	$e^{\frac{f(x)}{x}} \left(1 + f'(x) - \frac{f(x)}{x}\right) = 0$
Simplify	$xy' - y + x = 0$

Input

BROWN

Data Generation

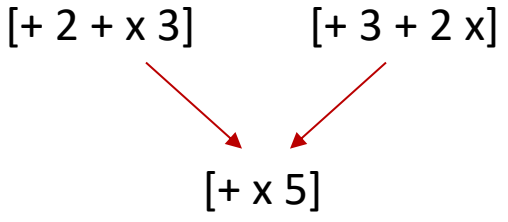
Task 3 : Second-order ODEs (ODE 2)

	output
Generate a random function	$f(x) = c_1 e^x + c_2 e^{-x}$
Solve in c_2	$c_2 = f(x)e^x - c_1 e^{2x} = F(x, f(x), c_1)$
Differentiate in x	$e^x\big(f'(x) + f(x)\big) - 2c_1 e^{2x} = 0$
Solve in c_1	$c_1 = \frac{1}{2}e^{-x}\big(f'(x) + f(x)\big) = G(x, f(x), f'(x))$
Differentiate in x	$0 = \frac{1}{2}e^{-x}\big(f''(x) - f(x)\big)$
Simplify	$y'' - y = 0$
	Input

BROWN

Data Cleaning

- Equation Simplification



- Coefficients Simplification

$$\log(x^2) + c\log(x) \rightarrow c\log(x)$$

- Invalid Expressions

$$\log(0) \quad \sqrt{-2}$$

BROWN

Experiment

- expressions with up to $n = 15$ internal nodes
- $L = 11$ leaf values in $\{x\} \cup \{-5, \ldots, 5\} \setminus \{0\}$
- $p_2 = 4$ binary operators: $+, -, \times, /$
- $p_1 = 15$ unary operators: exp, log, sqrt, sin, cos, tan, $\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$, sinh, cosh, tanh, $\sinh^{-1}$, $\cosh^{-1}$, $\tanh^{-1}$

The total number of trees satisfying this condition is around 10^{35}.
After data cleaning, this number could be reduced.

BROWN

Experiment

	Forward	Backward	Integration by parts	ODE 1	ODE 2
Training set size	20M	40M	20M	40M	40M
Input length	18.9±6.9	70.2±47.8	17.5±9.1	123.6±115.7	149.1±130.2
Output length	49.6±48.3	21.3±8.3	26.4±11.3	23.0±15.2	24.3±14.9
Length ratio	2.7	0.4	2.0	0.4	0.1
Input max length	69	450	226	508	508
Output max length	508	75	206	474	335

Table 1: **Training set sizes and length of expressions (in tokens) for different datasets.** `FWD` and `IBP` tend to generate examples with outputs much longer than the inputs, while the `BWD` approach generates shorter outputs. Like in the `BWD` case, ODE generators tend to produce solutions much shorter than their equations.

BROWN

Experiment

Hyperparameters:

- A transformer model with 8 attention head, 6 layers, and a width of 512 is applied.
- Use Adam optimizer with learning rate 0.0001.
- Remove sequences with more than 512 tokens.
- Batch size chosen to be 256.
- Beam size chosen to be 1, 10, 50.

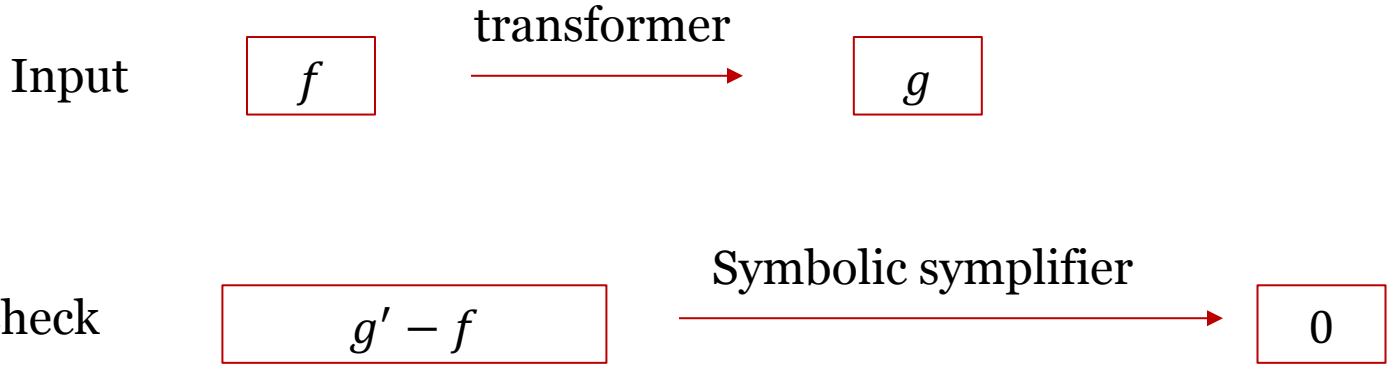
Loss function is chosen to be the standard log-likelihood score during training.

During test, the percentage of predictions that are correct is calculated.

BROWN

Experiment

How to determine whether a prediction is correct or not?



BROWN

Experiment

	Integration (FWD)	Integration (BWD)	Integration (IBP)	ODE (order 1)	ODE (order 2)
Beam size 1	93.6	98.4	96.8	77.6	43.0
Beam size 10	95.6	99.4	99.2	90.5	73.0
Beam size 50	96.2	99.7	99.5	94.0	81.2

Table 2: **Accuracy of our models on integration and differential equation solving.** Results are reported on a held out test set of 5000 equations. For differential equations, using beam search decoding significantly improves the accuracy of the model.

	Integration (BWD)	ODE (order 1)	ODE (order 2)
Mathematica (30s)	84.0	77.2	61.6
Matlab	65.2	-	-
Maple	67.4	-	-
Beam size 1	98.4	81.2	40.8
Beam size 10	99.6	94.0	73.2
Beam size 50	99.6	97.0	81.0

Table 3: **Comparison of our model with Mathematica, Maple and Matlab on a test set of 500 equations.** For Mathematica we report results by setting a timeout of 30 seconds per equation. On a given equation, our model typically finds the solution in less than a second.

Experiment

Transformer beat Mathematica?

Equation	Solution
$y' = \frac{16x^3 - 42x^2 + 2x}{(-16x^8 + 112x^7 - 204x^6 + 28x^5 - x^4 + 1)^{1/2}}$	$y = \sin^{-1}(4x^4 - 14x^3 + x^2)$
$3xy\cos(x) - \sqrt{9x^2\sin(x)^2 + 1}y' + 3y\sin(x) = 0$	$y = c\exp\left(\sinh^{-1}(3x\sin(x))\right)$
$4x^4yy'' - 8x^4y'^2 - 8x^3yy' - 3x^3y'' - 8x^2y^2 - 6x^2y' - 3x^2y'' - 9xy' - 3y = 0$	$y = \frac{c_1 + 3x + 3\log(x)}{x(c_2 + 4x)}$

Table 4: Examples of problems that our model is able to solve, on which Mathematica and Matlab were not able to find a solution. For each equation, our model finds a valid solution with greedy decoding.

BROWN

Experiment

Transformer can generate equivalent results of different forms

Hypothesis	Score	Hypothesis	Score
$\frac{9\sqrt{x}\sqrt{\frac{1}{\log(x)}}}{\sqrt{c+2x}}$	-0.047	$\frac{9}{\sqrt{\frac{c\log(x)}{x}+2\log(x)}}$	-0.124
$\frac{9\sqrt{x}}{\sqrt{c+2x}\sqrt{\log(x)}}$	-0.056	$\frac{9\sqrt{x}}{\sqrt{c\log(x)+2x\log(x)}}$	-0.139
$\frac{9\sqrt{2}\sqrt{x}\sqrt{\frac{1}{\log(x)}}}{2\sqrt{c+x}}$	-0.115	$\frac{9}{\sqrt{\frac{c}{x}+2}\sqrt{\log(x)}}$	-0.144
$9\sqrt{x}\sqrt{\frac{1}{c\log(x)+2x\log(x)}}$	-0.117	$9\sqrt{\frac{1}{\frac{c\log(x)}{x}+2\log(x)}}$	-0.205
$\frac{9\sqrt{2}\sqrt{x}}{2\sqrt{c+x}\sqrt{\log(x)}}$	-0.124	$9\sqrt{x}\sqrt{\frac{1}{c\log(x)+2x\log(x)+\log(x)}}$	-0.232

Table 5: Top 10 generations of our model for the first order differential equation $162x\log(x)y' + 2y^3\log(x)^2 - 81y\log(x) + 81y = 0$, generated with a beam search. All hypotheses are valid solutions, and are equivalent up to a change of the variable c. Scores are log-probabilities normalized by sequence lengths.

BROWN

Experiment

Generalization across generators

Training data	Forward (FWD)			Backward (BWD)			Integration by parts (IBP)		
	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50
FWD	93.6	95.6	96.2	10.9	13.9	17.2	85.6	86.8	88.9
BWD	18.9	24.6	27.5	98.4	99.4	99.7	42.9	54.6	59.2
BWD + IBP	41.6	54.9	56.1	98.2	99.4	99.7	96.8	99.2	99.5
BWD + IBP + FWD	89.1	93.4	94.3	98.1	99.3	99.7	97.2	99.4	99.7

BROWN

Limitations

- The dataset
  - The functions are simple: the coefficients range in {-5, -4, … , 5}. Could not generalize to the more sophisticated scenario.
  - If we want real-world applications, the cost of getting these data could be huge.
- The model
  - Highly relies on existing symbolic packages
  - Hardly learning something ‘new’ – It is more like memorizing old staffs

Advantages and possible extensions

- Advantages
  - Faster prediction
  - Provide a different view of looking at symbolic math problems
- Future research:
  - Neural network pruning?
  - Formula generator?
  - Incorporate special function?
  - Reinforcement learning?
  - Semisupervised learning?

within framework

beyond framework

BROWN

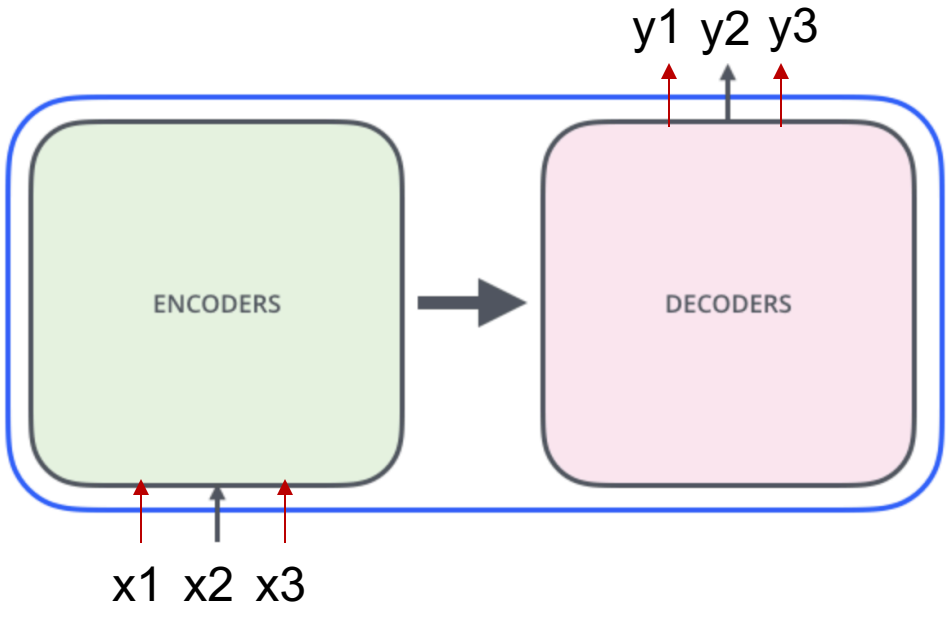
Conclusion

[This] transformer model … can perform extremely well both at computing function integrals and and solving differential equations, outperforming … MATLAB or Mathematica

The transformer model outperforms Mathematica and MATLAB in computing symbolic indefinite integrals of enormously complex functions of a single variable ‘x’ whose integral is a much smaller elementary function containing no constant symbols other than the integers −5 to 5.

BROWN

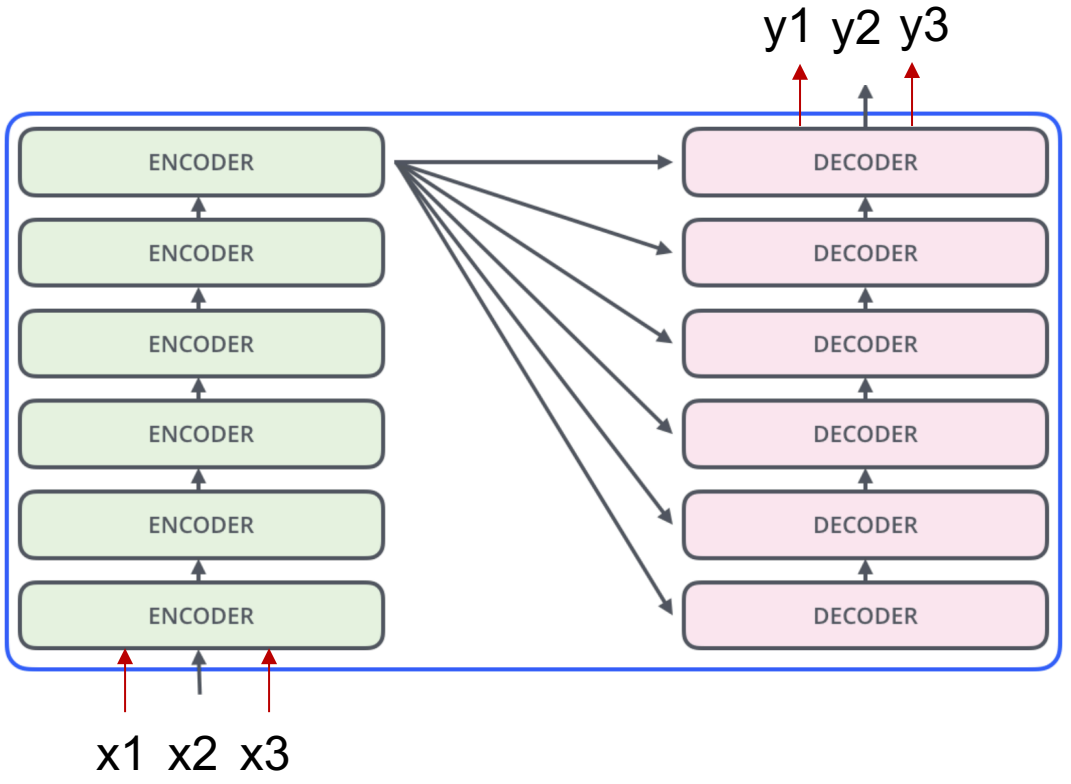
Transformer



The illustrated transformer, http://jalammar.github.io/illustrated-transformer/

BROWN

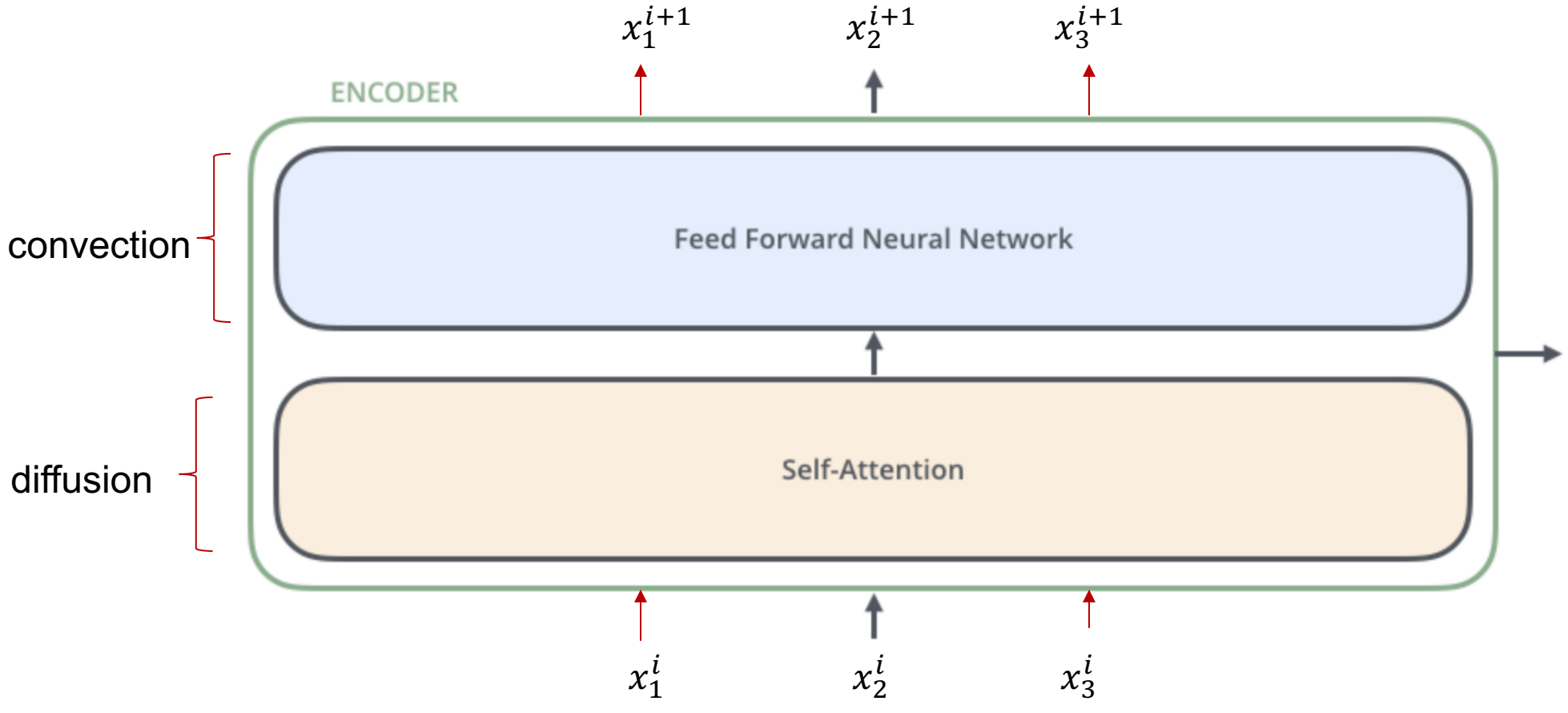
Transformer



The illustrated transformer, http://jalammar.github.io/illustrated-transformer/

BROWN

Transformer



The illustrated transformer, http://jalammar.github.io/illustrated-transformer/

BROWN

Transformer

Self Attention

$$\tilde{x}_{l,i} = x_{l,i} + \text{Concat}\left(\text{head}_1, \ldots, \text{head}_H\right) W^{O,l},$$

$$\text{where head}_k = \sum_{j=1}^{n} \alpha_{ij}^{(k)} [x_{l,j} W_k^{V,l}] = \sum_{j=1}^{n} \left(\frac{\exp(e_{ij}^{(k)})}{\sum_{q=1}^{n} \exp(e_{iq}^{(k)})} \right) [x_{l,j} W_k^{V,l}],$$

and $e_{ij}^{(k)}$ is computed as the dot product of input $x_{l,i}$ and $x_{l,j}$ with linear projection matrices $W_k^{Q,l}$ and $W_k^{K,l}$, i.e., $e_{ij}^{(k)} = d_{model}^{-1/2} \cdot (x_{l,i} W_k^{Q,l})(x_{l,j} W_k^{K,l})^T$.

Feed forward neural network $x_{l+1,i} = \tilde{x}_{l,i} + \text{FFN}_{W_{ffn}^l}(\tilde{x}_{l,i}),$

BROWN

Thank you.

BROWN