Hi! I am Prathamesh and I’ll be explaining my work briefly in this doc.

The task

(2*a-31)*(7*a-25)=14*a**2-267*a+775 - The dataset consisted of such expressions separated by an ‘=’ sign and my task was to predict the right expression given the left expression (using DL, no algorithms)

A few things about the dataset:

- The dataset only consists of:-
  - Alphabets - a, c, h, i, j, k, n, o, s, t, x, y, z
  - Numbers - 0 to 9
  - Trigo - Sin, cos, tan
  - Symbols - (,), +, -, *, **

Key observation: *Only a random set of letters are present in train data.*

Handling of unknown tokens!

- **Missing letters:** Only 13 random set of letters are being used in the training set. To solve this problem and make it work for 26 letters I thought of 3 methods!
  - <u>Data augmentation:</u> Once we separate the expression into different elements we can replace the letter variable with other letters on both sides of the equation and train on this augmented data.
  - <u>Replacement (Hard):</u> Since all expressions in the train set have only one variable per expression, we can simply replace the variable with a single token every time. i.e. replace any variable letter like a,b,c that we encounter with a single token like ‘x’ and re-replace it when outputting the prediction. This will significantly help the model since it can now allocate all of its power to learn the maths.
  - <u>Replacement (Soft):</u> Replace all unknown variables with a random known variable, get prediction from the model and re-replace it with original variable. **Since I was told not to use algorithmic methods, just to be safe, I used this method to handle the unknowns, but in a real setting method 2 would be my choice.**

A few other observations

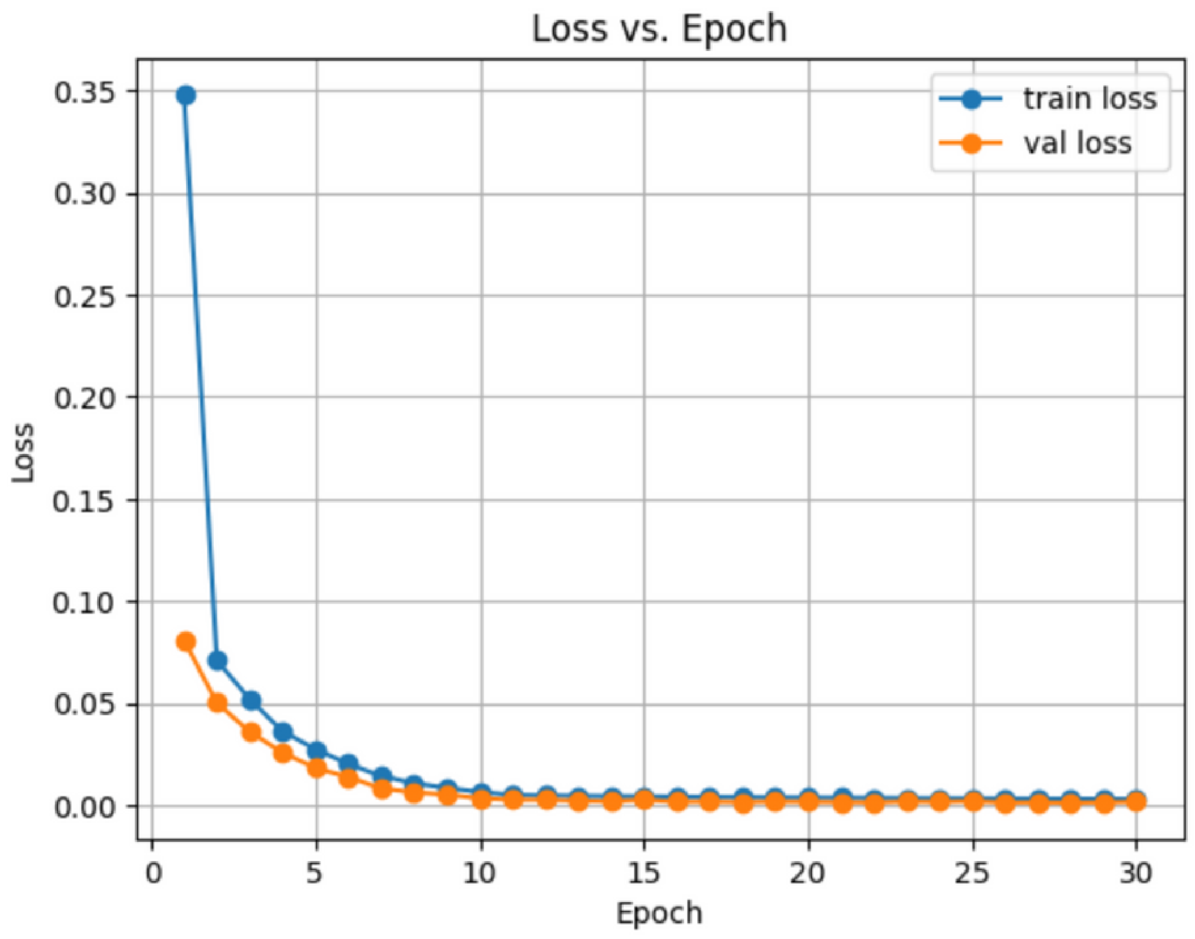
- In the trainset all numbers are small, hence the model fails to work for bigger numbers.
- The trainset only contains expressions with degree 2 polynomials and lower.
- The trainset only has whole numbers, no fractions.
- Following up on the last point, the trainset also doesnt have a division (/) operator to avoid fractional numbers.
- Other trignometric functions like sec(), cosec(), cot() aren’t present.
- Only “()” these type of brackets are present in the train set.

All of the above mentioned functionalities can be added by using an algorithmic solver to construct a more inclusive dataset, o selectively augmenting certain parts of the data (like switching types of brackets, trignometric functions, etc.)

My best model

Validation accuracy = 98.4% on 25000 held out samples

Training cycle loss graph

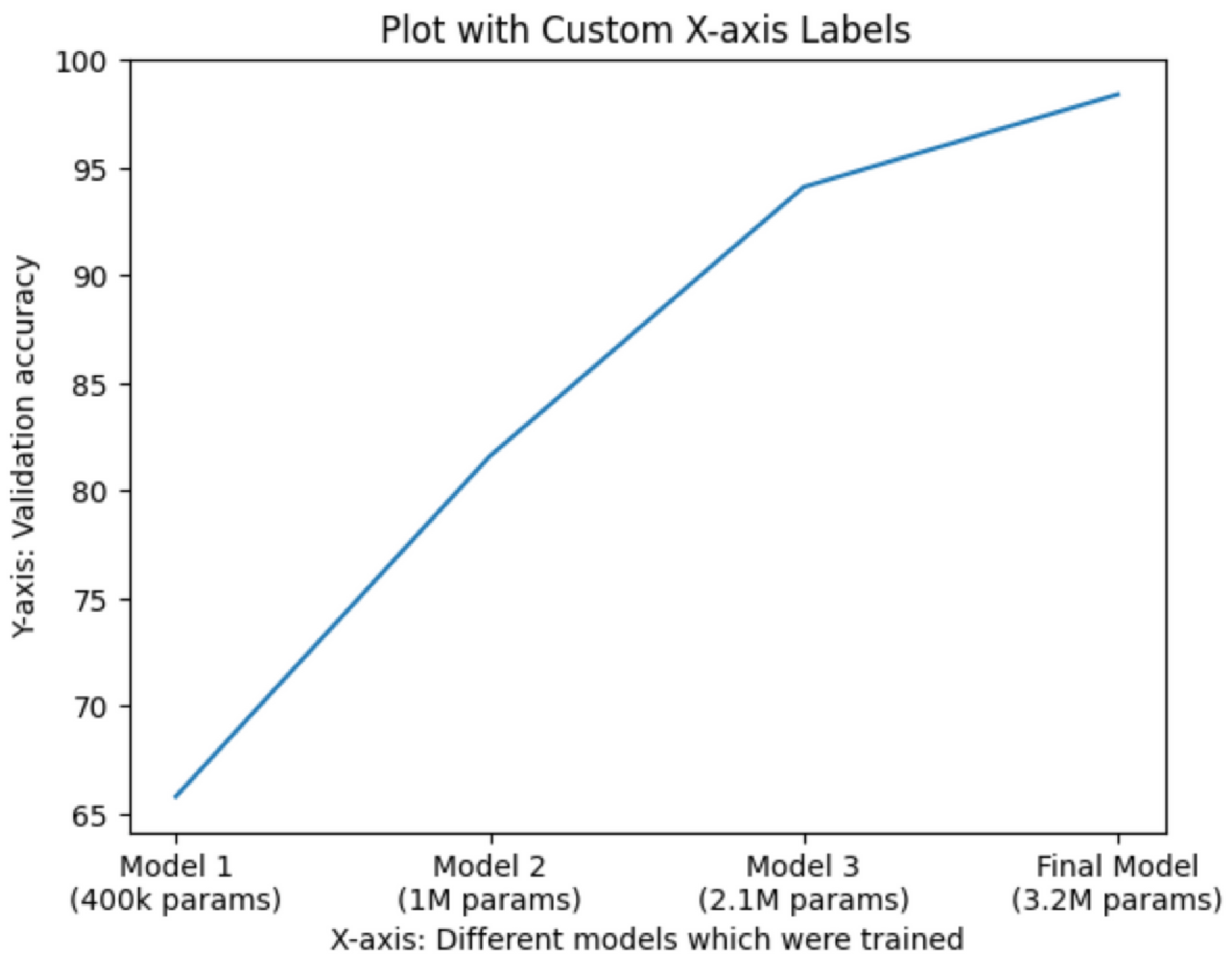



```
==========================================================================================
Layer (type:depth-idx)                        Param #
==========================================================================================
T5custom                                      --
├─Embedding: 1-1                              4,608
├─Embedding: 1-2                              4,096
├─T5Encoder: 1-3                              --
│    └─Embedding: 2-1                         4,608
│    └─Embedding: 2-2                         4,096
│    └─ModuleList: 2-3                        --
│    │    └─ModuleList: 3-1                   394,496
│    │    └─ModuleList: 3-2                   394,496
│    │    └─ModuleList: 3-3                   394,496
│    └─T5LayerNorm: 2-4                       128
├─T5Decoder: 1-4                              --
│    └─Embedding: 2-5                         4,608
│    └─Embedding: 2-6                         4,096
│    └─ModuleList: 2-7                        --
│    │    └─ModuleList: 3-4                   656,896
│    │    └─ModuleList: 3-5                   656,896
│    │    └─ModuleList: 3-6                   656,896
│    └─T5LayerNorm: 2-8                       128
├─Linear: 1-5                                 4,644
==========================================================================================
Total params: 3,185,188
Trainable params: 3,185,188
Non-trainable params: 0
==========================================================================================
```

Since I got 98.4% on my validation set and the lower bound given to me was 70% on test set, I also constructed other model architectures to see how low I could go in terms of model size while getting 70+ on validation set.

Some key features of my best models are:

- An encoder-decoder style architecture
- Multi-headed attention
- Cross-Attention
- Pre layer norm
- Position + token embeddings
- Tied token embeddings for encoder and Decoder.



References:

- Pytorch documentation
- Attention is all you need (https://arxiv.org/pdf/1706.03762)
- Text to Text transformer paper (https://arxiv.org/pdf/1910.10683.pdf)