Model architecture:

I used the Seq2Seq model to solve this challenge. Because both of our input and output are the math sequences, which lengths are likely unequal. Thus, we can't use the classification model which always have a limited range of labels for different inputs. This task is very similar to machine translation task, just different on transformation logic. And it is proved that Seq2Seq model solves the machine translation tasks successfully today.

As is known, text is kind of time sequential data. RNN is proved to have good performance on this time series data, but it will have problems like gradients disappearance when the length of input is large. Therefore, I used the improved model LSTM to build the model. Also, we can use other improved RNN models such as GRU.

According to Seq2Seq thoughts, I used two LSTM layers for encoder and decoder. And I used embedding layer before each LSTM to embed words into low dimension dense words embedding space. For the final layer, I used Dense layer with the activation function softmax to infer the specific output word with the maximum probability.

Hyper parameter tuning:

We have already known that the maximum length of out input is 29, and I have counted all data and found the number of unique words is 32. Besides, I have to use three special words to indicate start(<bos>), end(<eos>) and padding(<pad>) for sequence. Thus, I set max_length=32 and vocab_size=35.

Normally, the hidden dimension size for LSTM is set to 128, 256, 512. I don't think this task needs very complex hidden dimension greater than 256 to represent, so I tried 128, 256 and decided to use 256 for hidden dimension size. As for batch size, I believe 512 or larger size is suitable for the task considering about our 1M dataset. I have tried 512 for batch size and it needed more than 1000 steps per epoch, so I think batch size 1024 is suitable. For the first time, I just set epochs to 1 and saw what the loss and accuracy was like. The accuracy and loss was about 0.8 and 0.4 respectively, so it seemed model needs more epochs to converge. Thus, I just set epochs to 10 finally. As for learning rate, I just set it to common 0.005. All the parameters can be changed according to user's command.

Model performance on test cases:

Test Accuracy: 0.9859623312950134
Test Loss: 0.038746245205402374

References:

1. https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
2. https://medium.com/deep-learning-with-keras/seq2seq-part-e-encoder-decoder-for-variable-input-output-size-with-teacher-forcing-92c476dd9b0