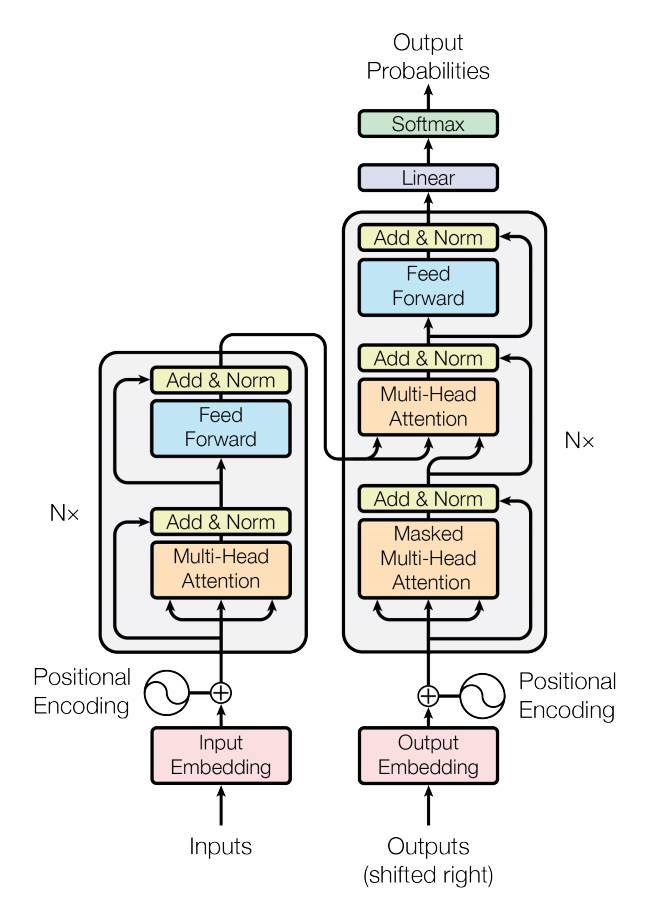
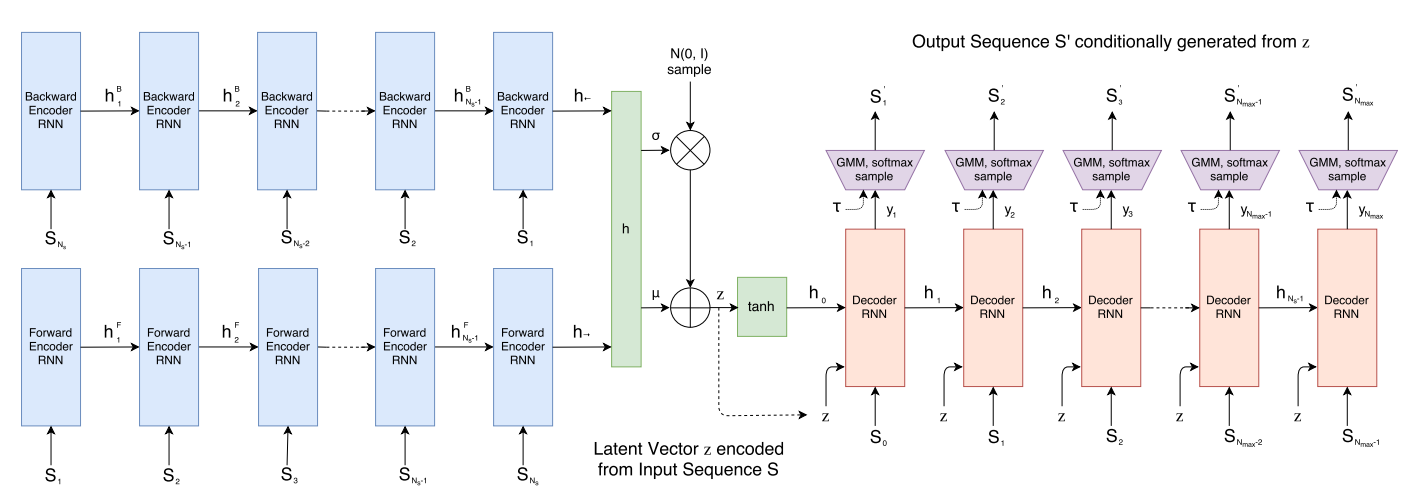
# Doodle Generation

## Implemented baselines

So, far we have implemented 2 baseline models.

1. **[Model 1]** A simplified version of the Google SketchRNN model  
   To understand the motivation behind why that complex model is expected to work.  
   
2. **[SketchRNN]** A ground-up replica of the Google SketchRNN model  
   To begin the project objectively.  
   

## Challenges we faced

1. Pre-processing data for [Model 1] was time-consuming. We had to make decisions regarding the input shape, the training process and the expected output of our model.
   1. Converted **Doodle<Stroke<x[], y[], timestamp[]>[]>[]** into **Doodle<Stroke<Coordinate<x, y>[]>[]>[]**
   2. Decided to drop the timestamp information as it is not relevant to our project.
2. [Model 1] did not show loss reductions per epoch. It justified that we needed a more complex model to learn the structure of our data, however, for a very long time, we did not have a running model and only analyzed our failures.
3. [SketchRNN] is still in implementation as on Monday (Apr 15). The results of running this model are not analysed yet.
4. Colab does not inherently support user input within the notebook. This makes human evaluation very difficult, without HTML embedding, which took time to write.

## What would we like to complete by the final report?

1. Maximize performance out of [SketchRNN] model, and prep it for evaluation against an objective metric rather than human evaluation.
2. Implement some evaluation metrics like GAN (discriminator model) - a novelty to Google paper.

## Preliminary result

Baseline model implemented

## Preliminary analysis

1. Just positional encoding does not work as it does not seem to learn the structure of our data, even though our data implicitly stores [x,y] coordinates for every stroke.
2. Just a linear deep model does not work as position data is essential. It is similar to the case study presented in the lecture, where training a vanilla NN on image data is the same as training the model on noisy images, as it does not understand locality. Hence, attention is what we need, indeed!
3. This analysis concludes that we need an RNN/CNN component along with the transformer encoder-decoder to learn on our dataset of doodles so that the model can capture the structure as well as locality (attention) when it comes to drawing strokes.

## Any changes in scope or project direction? - No