# Assignment 2: Neural Machine Translation

**Prateek Sachan**
M.Tech (CSA)
SR: 15754
prateeksachan@iisc.ac.in

## Abstract

Neural Machine Translation from English to German and Hindi language using Seq2Seq with different attention models.

## 1 Preprocessing

In pre-processing step, I removed all the pancutation characters and ignored all the words that appear less then 5 times in the whole corpus for both input and target language.

## 2 Experimental Setup

Code is written using Pytorch. Used fasttext as pretrained embedding and didnt allowed word embedding training to reduce the number of parameters to train. Start and End token are added to all input and target string.

## 3 Seq2Seq Model

Single layer LSTM based seq2seq model was implmented as described in the class. Encoder takes 300d fasttext embedding of a word at each time step. LSTM unit then generate 128d hidden and cell state for next time step. Final hidden state of the encoder is given to the decoder as initial hidden at first time step along with start token. Output from decoder LSTM is feeded to a Linear classifier with a softmax layer that gives probability of next word from the whole vocabulary. While training correct next word is feeded at every time step to decoder and cross entropy loss is calculated. Final loss is sum of all the losses at each time step. I also tried same experimentation using GRU in place of LSTM.

## 4 Attention

I implmented dot product, additive and multiplicative attention. In dot product attention, at each decoder time step, dot product of all the encoder hiddent state and decoder hidden state gives the attention score. Attention layer output is then the weighted sum of all the encoder output states where weights are the attention scores. These attention outputs are concatenated to the decoder output for prediction of next word.

In multiplicative attention, a new weight matrix $W$ is used which is set to trainable. Attention score in multiplicative attention is calculated as $H * W * h$, where $H$ is a matrix that contains all the hidden states if encoder and $h$ is decoder hidden state. Now these scores are used in similar way as in dot product attention.

In additive attention, 2 new weight matrix $W_1$, $W_2$ and a vector $v$ is used which are set to trainable. Attention score in additive attention is calculated as $v * tanh(W_1 * H + W_2 * h)$, where $H$ is a matrix that contains all the hidden states if encoder and $h$ is decoder hidden state. Attention scores are used in similar way as in dot product attention.

## 5 Results

Result of all the model were very bad, getting blue score of 1 or 2 maximum on test data. I tried to find the reasons for low blue score but not able to make final model to work correctly. I tried with different vocabulary size, making embeddings trainable, chainging preprocessing methods and tried to debug code at my best till the last day but still results were not upto the mark. This is the reason I did not implemented key value and self attention.