# CIS590/890 DL - Assignment 5

## 1 Goal

In this assignment, you will train and evaluate RNN, LSTM and GRU models, on two tasks: 1) sentiment classification (specifically, predicting the sentiment of a short text), and 2) language modeling (predicting the next word in a text sequence). The goal is to gain experience with recurrent neural networks for sequence classification and generation.

## 2 Dataset

You will use The Stanford Sentiment Treebank (SST-5) dataset to train and test your models. The SST-5 dataset was published by Socher et al. (2014), and it's available at: https://nlp.stanford.edu/sentiment/. It consists of 11,855 sentences extracted from movie reviews with one of five fine-grained sentiment labels: 1 (strongly negative), 2 (weakly negative), 3 (neutral), 4 (weakly positive), 5 (strongly positive). We converted the original tree-structure of the data to raw text (together with the corresponding class labels) using the pytreebank library. The resulting train/test/dev datasets are available on Canvas.

## 3 Sentiment Classification

Perform the following tasks:

1. **Load and preprocess the data:**
   - You will use the char-level representation of text in your RNN models. Specifically, use the extended ASCII with 256 characters, and represent each character in a sentence as a one-hot-encoding.
2. **Implement RNN, LSTM, and GRU models:**
   - **Create the Model/Network:** Define models for vanilla RNN, LSTM and GRU networks, respectively, using the char-level representation.

- **Train the Model:** Train your model using the sentiment train subset. To get the sentence encoding, use the element-wise max of all hidden states.

3. **Experimentation and Analysis** Complete the following analysis, and include your findings in the report.

    - Periodically compute the loss on the validation set, and create a plot with the training and validation loss as training progresses. Is your model overfitting? Include the plot in your report. Optional: Try dropout if your model is overfitting.
    - Experiment with the learning rate. You can try a few different learning rates and observe how this affects the loss. A common practice is to drop the learning rate when the loss has plateaued. Observe the loss plots, and investigate the need for lowering the learning rate. If needed, lower the learning rate and show the effect on the loss.
    - Experiment with the size of the hidden layer or the model architecture. How does doubling the size of the hidden layer affect the validation accuracy?
    - For each RNN model architecture, identify the best model based on the accuracy on the dev subset.
    - For one type of RNN architecture, experiment with a bidirectional network and compare the results with the resulting from the corresponding unidirectional network.

4. **Evaluate the models:** Evaluate your best vanilla RNN, LSTM and GRU models on the test data and compare the results of the three models. Use precision, recall, and F1-measure to report the performance on the test set. Compare the results of different types of RNN architectures: vanilla RNN, LSTM, GRU. Offer your intuition behind any observed difference in performance between the models. For the best overall model, discuss where your model is making mistakes. Use a confusion matrix plot to support your answer.

## 4 Text generation using char-rnn

In this task, you will use the SST-5 dataset to build a language model using an architecture of your choice (vanilla RNN, LSTM or GRU). You will notice that the code is quite similar to that of the classification problem. The biggest difference is in the loss function. For classification, we run the entire sequence through the RNN and then impose a loss only on the final class prediction. For the text generation task, we impose a loss at each step of the RNN on the predicted character. The classes in this second task are the possible characters to predict.

Show samples of the text generated by your model at an early stage of the training (e.g., after 100 epochs), at an intermediate stage (e.g., after 500 epochs) and at a later stage (e.g., after 1000 epochs). Give a qualitative discussion of the results. Where does the model do well? Where does it seem to fail?

## 5 What to submit

Submit a Jupiter Notebook containing your code and results/discussion, or python code together with a report file showing the results/discussion.

## 6 Useful Links

Links to sample code that you may find useful are provided below (and are also available on Canvas):

- Padding/masking sequences in a mini-batch in PyTorch:
  - `https://towardsdatascience.com/taming-lstms-variable-sized-mini-batches-and-why-py`
- Padding/masking sequences of different length in TensorFlow/Keras:
  - `https://www.tensorflow.org/guide/keras/masking_and_padding`
- Char-level RNN for text classification in PyTorch:
  - `https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html`
  - `https://github.com/spro/practical-pytorch/tree/master/char-rnn-classification/char-rnn-classification.ipynb`
- Char-level RNN for text classification in TensorFlow:
  - `https://www.tensorflow.org/tutorials/text/text_classification_rnn`
- Char-level RNN for text generation in PyTorch:
  - `https://pytorch.org/tutorials/intermediate/char_rnn_generation_tutorial.html`
  - `https://github.com/spro/practical-pytorch/tree/master/char-rnn-generation`
- Char-level RNN for text classification in TensorFlow:
  - `https://www.tensorflow.org/tutorials/text/text_generation`