

## Deep Learning Practicum - Spring 2025

This final notebook has you train a model that classifies words as being in Pig Latin or English. Pig Latin is a "playful" way of speaking English. If the word starts with a consonant or consonant cluster (such as "sh" or "ch"), then that cluster is moved to the end of the word and the suffix "ay" is added to the word. so "changes" becomes "angeschay". If the word starts with a vowel then "way" is added as a suffix, so "appear" becomes "appearway".

**IMPORTANT:** Your final submission must be as a single PDF file, NOT a zipped directory of ipynb files, html files, etc that I then have to sift through. In other words, convert your marked-up ipynb file as a pdf file. This can be done by exporting the notebook as an html file, opening the html file in a browser and then printing the browser window to a pdf file. Submission is to be uploaded to your directory with the filename LASTNAME-FIRSTNAME-FINAL.pdf. Submissions that do not fit this format may not be graded. I will gather your submissions on the morning of April 30th.

1. The `pig_latin.txt` file is a text file. Each line of the file contains an english word separated from its pig-latin equivalent by a blank space. Use this file to create a numpy array, `X`, whose components are words (English and Pig-Latin) and another numpy array, `Y`, whose components are the targets 0 if the word is English and 1 if it is Pig-Latin. Replace any hyphen ("-") in a word with a blank space. Remove single quotes and new line characters (`\n`) from the words. Print out the number of data samples you have.
2. Construct a *vocabulary* dictionary for the input samples (words) that maps each character of the words to a unique integer. Encode the blank space character, (" "), as 0. Print out the size of your vocabulary.
3. Build a function that encodes words as integer arrays using the vocabulary you created in the preceding part. Your encoder should return the integer encoded word as a numpy array of length 30. Use zero padding for words of length less than 30. Build a function that decodes an integer encoded word. Take the word 'resilience' and print out the encoded representation generated by your encoder. Print out the decoded word generated by your decoder.
4. Take your data arrays, `X` and `Y`, and form Tensorflow Dataset objects for the p-training, validation, and testing datasets. Be sure to shuffle these dataset objects. Assume the testing dataset contains 20% of the original data samples. Assume the validation dataset uses a 20% validation split on the remaining training samples. Use a batch size of 64. Determine how many batches are in each of the datasets.
5. Instantiate a Keras model that uses a single self-attention layer to classify the input sample as either an English or Pig-Latin word. Your model should take the integer encoded inputs from your dataset object and return 2 outputs representing the probability of the word being English or Pig-Latin. Print a summary of your model and identify the number of trainable parameters in your model.
6. Select a model that achieves at least 99.0 % validation accuracy. Adjust the model architecture and regularization until you achieve this objective. Plot the training curves that gave you your best model and determine the test accuracy of the best model.
7. For the best model you determined in the preceding section, evaluate the model's f1-score for thresholds that are regularly spaced at an interval of 0.01 starting from 0.01 and going to 1.0. Plot the model's f1-scores versus threshold, determine the threshold that gives the largest f1-score. Print out that threshold and that f1-score.