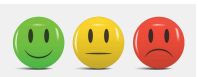
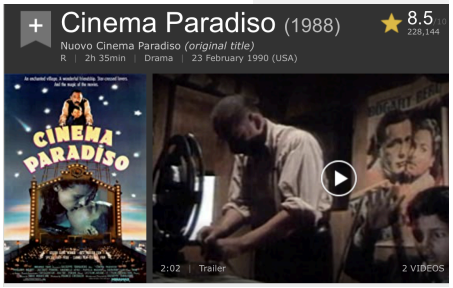
Introduction to Deep Learning - Exercise #3

submission date: 18/07/2021

Programing Task: Sentiment Analysis for the IMDB Movies Reviews Dataset





The dataset consists of 50,000 annotated reviews. Each review is a plain text describing the viewers' experience and opinion on the movie watched. The dataset consists of highly polar reviews and contains binary labels (positive and negative) based on the number of stars the movie received by the viewers. Some reviews are very long, but we will consider only the first 100 words they contain.

In this exercise we design networks that predict the viewers' sentiment (positive/negative) towards the movies they watched based on the review they wrote. We will compare the following four different strategies:

1. The use of a simple Elman RNN
2. The use of a simple GRU
3. The use of a global average pooling. In this case every word goes through an MLP that result in a scalar, *sub-prediction score*, and these scores are then summed together to provide the final prediction
4. Adding a local self-attention layer to Strategy #3 in order to achieve crossword reasoning (contextualization)

The exercise comes with a partial code that loads the reviews from the dataset, and processes them into a long list of lower-case text (100 words, no punctuation/sentences, no special characters). Longer reviews are truncated, shorter ones are padded. This pre-processing uses the GloVe word embedding which maps every word into a 100-dimensional vector. To avoid over-fitting over this small dataset we do not allow this embedding to train. You can read more about this embedding here. Some parts of these pre-processing steps were taken from this tutorial.

Specific tasks:

1. Fill in the missing lines of code in the RNN and GRU cells functions. The RNN contains some lines which you may find helpful (or choose to omit and implement on your own). The gates and update operators should consist of a single FC layer (hidden state dim. should be between 64-128 for the lowest test error). The convention of the tensors for these recurrent networks is: **batch element x "time" x feature vector**. So the recurrence (your iteration in the code) should apply on the second axis. Once the review is parsed, its hidden-state should pass through an MLP which produces the final output sentiment prediction (a 2-class one hot vector).

Run each of these two recurrent network architectures, describe your experiments with two different hidden-state dimensions, and the train/test accuracies obtained (with plots). Explain what could lead to the different results you found in the experiment (both RNN vs GRU and hidden state size). Come up with a test review of your own which demonstrates the different capabilities of the two recurrent models. Add this review (and maybe variations of it that you've experimented with), the results obtained and your explanation.

2. In a second experiment you will process each word by a small MLP to obtain the 2-class predictions (no softmax in this stage) over each word, which we term "*sub prediction scores*", and then sum up all these vectors to obtain a final prediction. Note that this averaging allows us to handle data of variable length.
   Aim for FC architecture that achieves a low test loss, and summarize it in your report. Since this model gives a score per word, output two test examples with these numbers next to their corresponding words (`reviews_text` and `sub_score` in the code). The examples should be selected such that one which the prediction is right and one which is wrong, for both Positive reviews and Negative reviews (TP, TN, FP, FN). Come up with an explanation of why the errors may have happened. To conduct this experiment you are welcome to write your own test reviews in loader.py in the `my_test_texts` list.

3. Write a *restricted* self-attention layer which queries every word with its closest 5 words on each side (using torch.roll and padding of size 5),add a positional encoding. This layer should have a single head and learnable query, key and value matrices. You can use the incomplete code in `ExLRestSelfAtten` to implement this layer or use your own code. This code can use torch.roll to shift the text right and left as well as torch.pad to handle the boundaries.

4. Finally, add this layer to the network architecture you used in Task 2 (before the sub prediction scores are computed) and repeat the experiments above, i.e., the four reviews (TP, TN, FP, FN). Print the per word sub prediction scores and check whether they changed. Explain the results (at least the improvement if you see one). In addition, create a review of your own where the context of a word changes given its neighboring word/s. See if you can get the network in Task 2 to fail on this example, while the new (self-atten.) network succeeds. Include the reviews, their per word scores and explanations in your report.

We are expecting you to report and elaborate on every practical task in the pdf, with your own words and analysis of what you've done. Include everything that you think is crucial for us to understand your way of thinking.

Theoretical Questions:

1. Explain what type of a network architecture you will use to handle each of the following problems (e.g., many-to-many RNN, or a one-to-one convolution NN). Explain your reasoning.
    a. Speech recognition (audio to text)
    b. Answer questions
    c. Sentiment analysis
    d. Image classification
    e. Single word translation

2. Text-to-Image:
    a. Describe the architecture of a network that reads a sentence and generates an image based on the text. Do not address the question of how such a network is trained, just explain why its architecture should have the capacity to perform this task. Assume that the images come from a restricted class of images, e.g., faces, and can be encoded, and decoded, in a low-dimensional latent space.

    b. Assume an image is encoded using 4 latent codes that correspond to its four quadrants. Explain how an attention can be used to allow such a network to better support fine-grained descriptions in the input text, which refers to the different regions (top, bottom, left, right, sky, ground, etc.). Note that the architecture should be able to link text to specific image code.

3. CNNs:
    a. Assume an 128X128x1 input image is inputted to the following architecture :

        conv(kernal_size=3,stride=2,padding=0)
        conv(kernal_size=5,stride=1,padding=2)
        conv(kernal_size=3,stride=1,padding=1)
        conv(kernal_size=5,stride=2,padding=0)

        What would be the output size of the response map produced by this network?

    b. What would be the size of the receptive field of each neuron in the final layer (consider center neurons which are not affected by the padding).

4. Transformer network:

Show that cascaded layers of self-attention, and FC (transformer we saw in class) give rise to a permutation-invariant operator (over the tokens). Show that by adding positional encoding, this ceases to be the case.

Submission Guidelines:

The submission is in **pairs.** Please submit a single zip file named “ex2_ID1_ID2.zip” where ID1 is the id number of one student and ID2 is of the other (e.g. ex2_123456789_987654321.zip). This file should contain your code, along with an ”ex2.pdf” file which should contain your answers to the theoretical part as well as the figures/text for the practical part. Furthermore, include in this compress file a README with your names and cse usernames.
Please write readable code, with documentation where needed, as the code will also be checked manually.
Late submission - 10 points reduction for each day. Submissions will not be accepted after 4 days.