IDL Exercise 3.

Daniel Ruderman, David Ponarovsky

April 2025

1 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)

   Solution. one-to-one CNN. The task requires iterating over each time-window and then mapping a wave function into its representation in the semantic space. Since, by the definition of the task, we should treat the signals as independent, we don't need to take into account influences of signals that we got at the beginning of the sentence over signals that we got at the end of the sentence.

   (b) Answer questions

   Solution. many-to-many RNN. Here, the task requires language processing, since the meaning of words might completely differ depending on the occurrence of other words, the inference process has to consider the sentence as a whole.

   (c) Sentiment analysis

   Solution. many-to-many RNN. In a similar manner to the above, sentiments depend on the whole context. The pair 'oh no' can refer to either anxiety or sarcasm, distinguishing the case requires access to the whole sentence.

   (d) Image classification

   Solution. one-to-one CNN. (or just a single CNN) Here, the dimensions of the image are (typically) defined and in some sense finite. Thus, it makes sense to consider the standard CNN that takes an image as input and performs the whole computation alone, and the mapping (one-to-one vs. many-to-many) is kind of degenerate itself.

   (e) Single word translation

   Solution. one-to-one CNN. (or just a single fully connected). In a manner similar to the above, since the mapping between languages, in the context of mapping single words, is approximately one-to-one,

and the dimension of the words is small and can be thought of as finite, a single fully connected net can be a good candidate for the task.

(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.

Solution. The net will be composed of tree parts:

- An Input Rnn layer, that gets the input and compute an segment output, on each we think as input that has a varity level of influences by all the words of the sentance.
- A many-to-many attention level that will take the intermediate output and then will 'mix' it.
- A CNN, that could be thought of as the inverse of the classification net.

The idea is that the first RNN should have the capacity to extract the meaning of the sentence (text processing task), and the second layer maps the entities to different 'locations' on the image with different weights.The thrid should have the capacity to output images based on the fed text, as it is the reverse of classifying images, a task at which CNNs excel.

(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 latent code.

Solution. Now we can take the architecture from earlier, and in addition, add an RNN layer that, given the sentence, understands the geometrical location, for example, 'left top corner' (ofcourse that the output would be a segement, with a score in each bit). Now one can concatenate that output with the intermediate output of the original first layer and give it to the attention layer. This, in fact, hints to the attention layer that it should focus on mapping to one of the four quadrants.

(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)
```

where `padding=0` is equivalent to `valid`. What would be the output size of the response map produced by this network?

Solution. We compute the dimension directly. We compute it by adding padding entries to each end, then moving the kernel window until we reach `last_entry - kernel_size`, and at the end, we divide by `stride` and append a single position of the kernel. So in overall, we get:

$$o_i = \frac{o_{i-1} + \textbf{pedding} * 2 - \textbf{kernelsize}}{\textbf{stride}} + 1$$
$$o_0 = \frac{128 + 0 * 2 - 3}{2} + 1 \to 63$$
$$o_1 = \frac{63 + 2 * 2 - 5}{1} + 1 \to 63$$
$$o_2 = \frac{63 + 1 * 2 - 3}{1} + 1 \to 63$$
$$o_3 = \frac{63 + 0 * 2 - 5}{2} + 1 \to 30$$

So the size of the response map is `30 x 30 x 1`.

(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).

Solution. Notice that the receptive field of one layer equals `kernel_size x stride`, and that the concatenation of layers increases the receptive field additively and the overall stride multiplicity. Therefore we get the follow recursive relation:

$$r_i = r_{i-1} + (\textbf{kernelsize}_i - 1) \cdot \prod_{j \le i} \textbf{stride}_j$$
$$r_0 = 1 + 3$$
$$r_1 = 3 + 2 * 4 = 11$$
$$r_2 = 11 + 2 * 2 = 15$$
$$r_3 = 15 + 2 * 4 = 23$$

2. Show and explain why a transformer composed solely of successive self-attention layers and fully connected (feed-forward) layers acts as a permutation-invariant function on its input tokens - i.e., its output is unchanged if the tokens are reordered. Then show that once positional encodings are added, this permutation invariance no longer holds.

Solution. Let's denote the softmax function by f, and W_Q, W_K, W_V as the learned weights for the queries, keys, and values. In addition, let's denote the normalization factor in the softmax (we can assume that the softmax is aware of the token dimension). Thus, the attention function becomes:

$$\text{Attention}\left(X\right)=f\left(\left(XW_Q\right)\left(XW_K\right)^\top\right)XW_V$$

So, under the action of permutation P, namely $X \to PX$, we find that the attention becomes:

$$\begin{aligned}\text{Attention}\left(PX\right)&=f\left(\left(PXW_Q\right)\left(PXW_K\right)^\top\right)PXW_V\\&=f\left(PXW_QW_K^\top X^\top P^\top\right)PXW_V\end{aligned}$$

Since f is a coordinate-wise function, and P is a permutation, we get that $f(PXP^\top)=Pf(X)P^\top$. In addition, using that P is a permutation again, we have that $P^\top P=I$. So overall, we get that:

$$\text{Attention}\left(PX\right)=P\text{Attention}\left(X\right)$$

So the Attention operator commutes with permutations, or in other words, is permutation-equivariant. Thus, permuting the input and then the labels entered into the fully connected net is equivalent to first computing the attention and then applying the permutation and its inverse.

Adding the positional encoding to the net breaks the commuting relation. We can formulate it by writing the input to the all connect as:

$$\text{Attention}\left(X\right)+g(I)$$

Where g is some positional function. Now, after permuting both the input and the labeling, we have:

$$P^\top\left(\text{Attention}\left(PX\right)+g(I)\right)=\text{Attention}\left(X\right)+P^\top g(I)$$

For g to make sense, there should be at least two indices such that $g_i \neq g_j$. Then, any permutation $P^\top$ that swaps between them is a permutation for which the net might be invariant.